C3. Réseaux de neurones multi-couches

Advanced Machine Learning (MLA)

Kévin Bailly et Bruno Gas

kevin.bailly@sorbonne-universite.fr

http://people.isir.upmc.fr/bailly/

Objectifs

- **Objectifs**
  - Comprendre l’optimisation des MLP par descente de gradient
  - Comprendre le graphe de calcul de la rétro-propagation du gradient
  - Savoir programmer un MLP sous Python

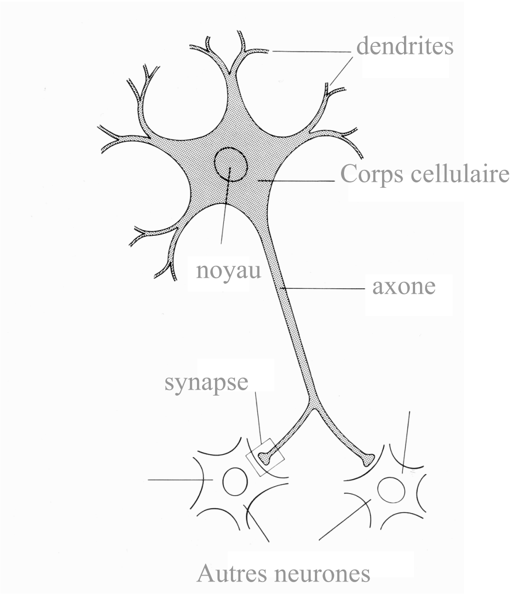
Plan

- **Plan**
  - Du neurone formel au perceptron
  - Apprentissage : perspective historique
  - L'optimisation par descente de gradient
  - Calcul du gradient
  - Perceptron multi-couches

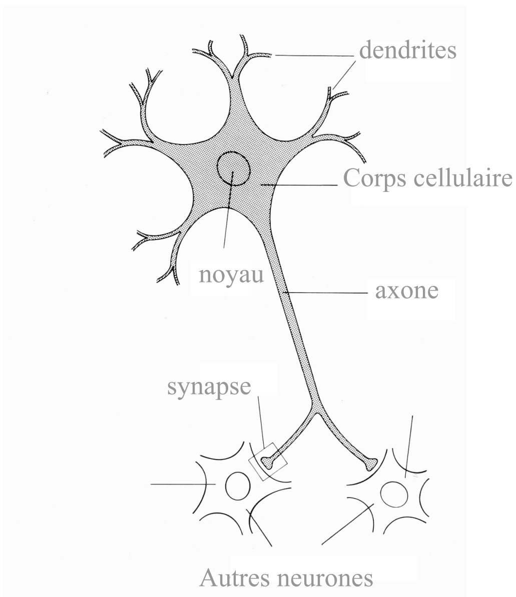
SCIENCES SORBONNE UNIVERSITÉ

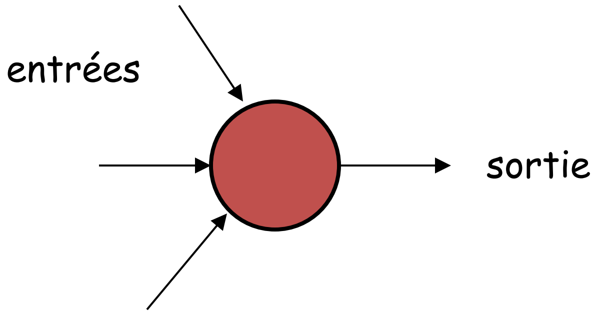
1.
Du neurone formel au perceptron

Le neurone formel



Le neurone formel

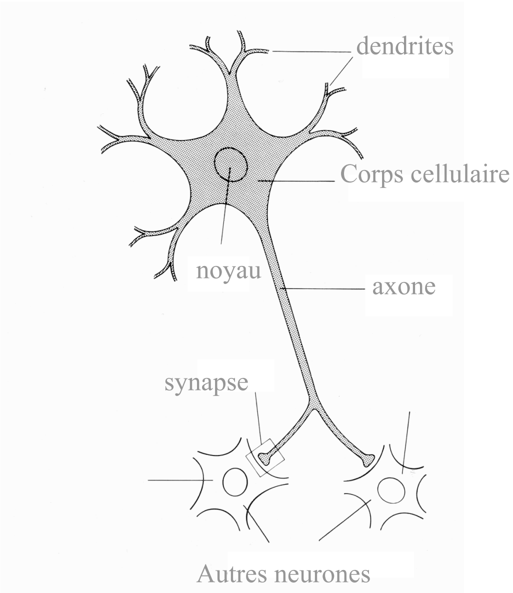


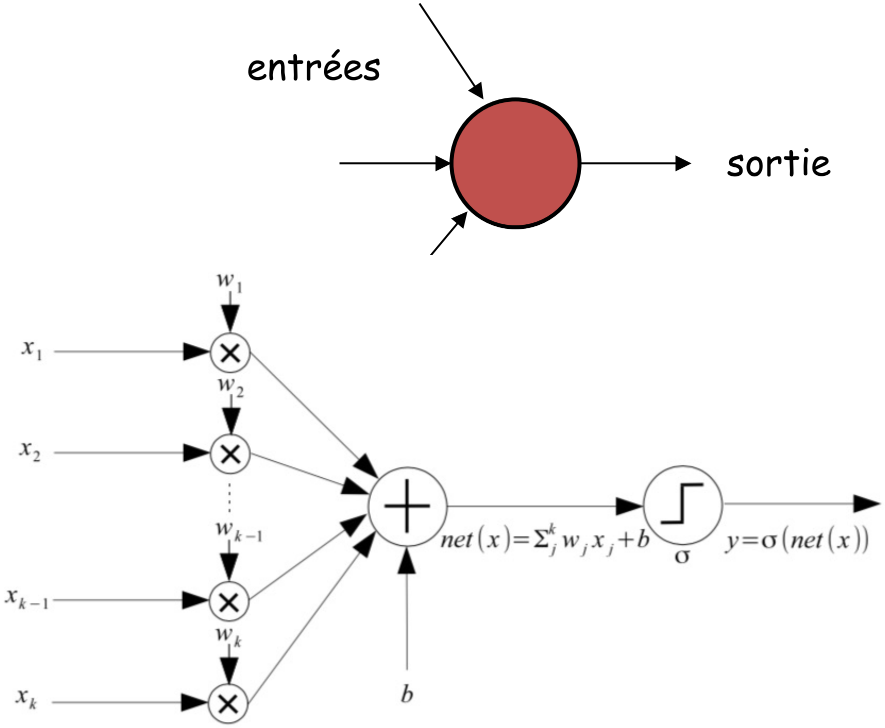


[McCullogh & Pitts, 1943]

> Permet de calculer n’importe qu’elle fonction si les poids sont bien choisis

Le neurone formel

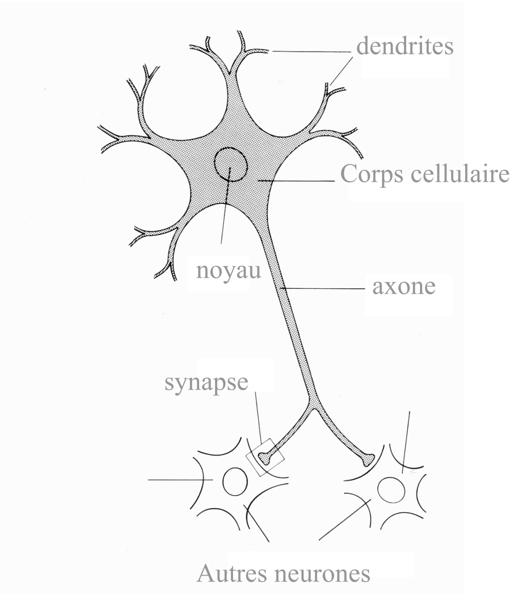


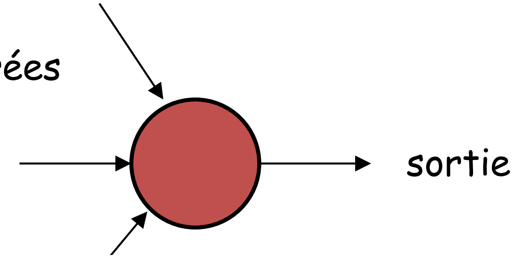


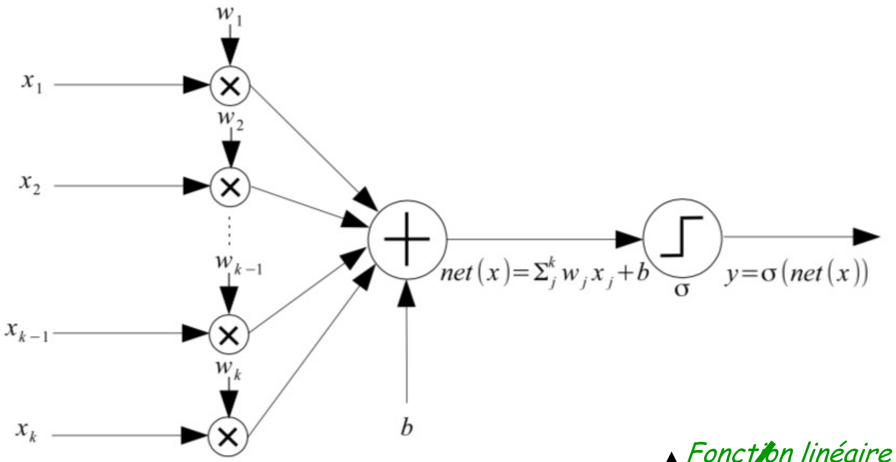
[McCullogh & Pitts, 1943]

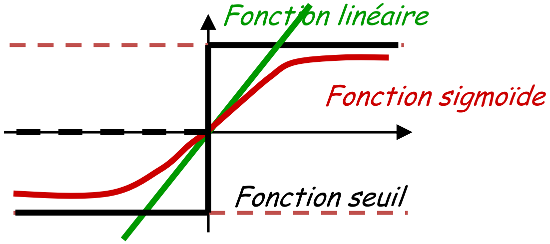
➢ Le neurone formel réalise une somme de ses entrées pondérées par les *poids synaptiques* avant seuillage par une fonction de transition

Le neurone formel





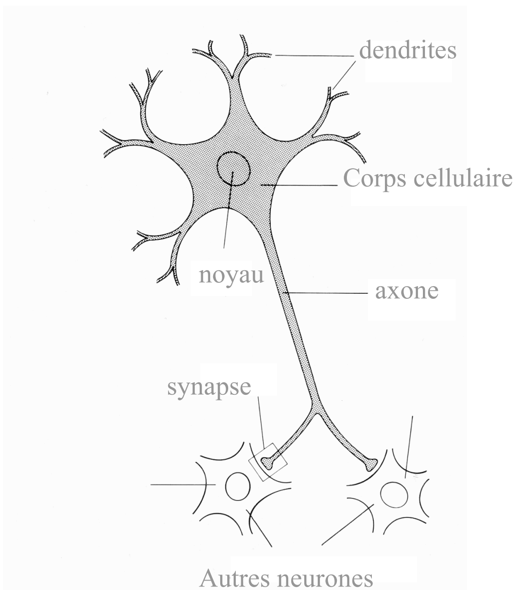


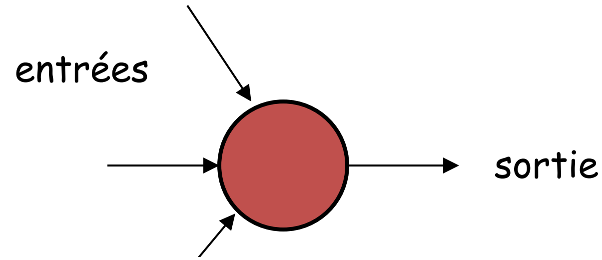


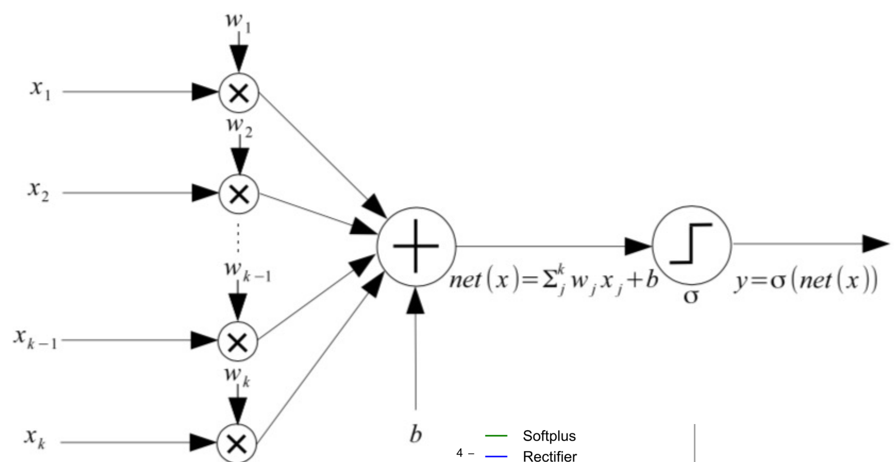
[McCullogh & Pitts, 1943]

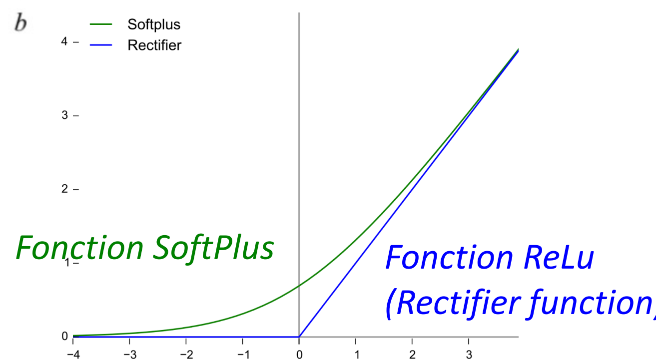
> Différentes fonctions de transition

Le neurone formel





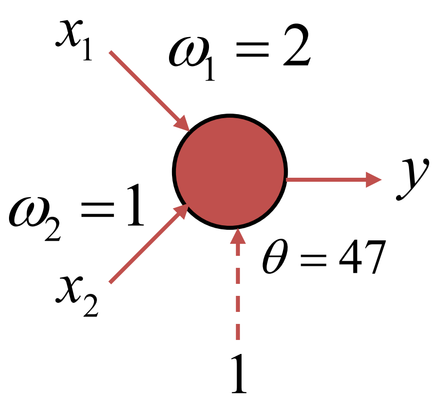




[McCullogh & Pitts, 1943]

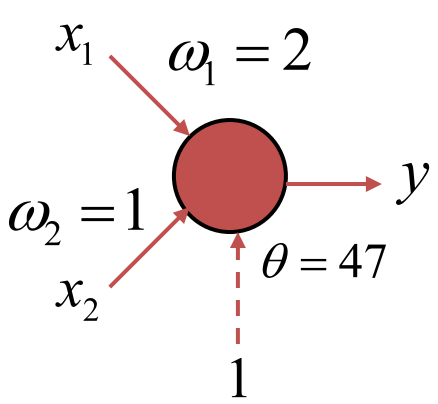
> Différentes fonctions de transition

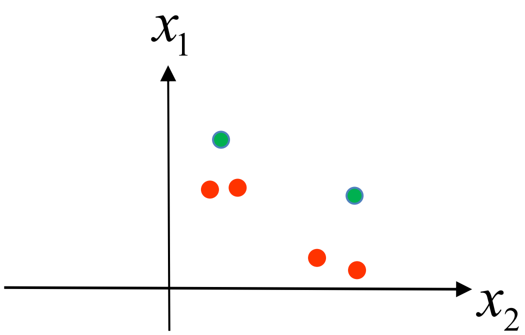
Interprétation géométrique



x_1	x_2	$2x_1 + x_2 - \theta$	y
20	8	48 - 47 = 1	1
15	20	50 - 47 = 3	1
16	10	42 - 47 = -5	-1
5	15	25 - 47 = -18	-1
16	6	38 - 47 = -9	-1
2	20	24 – 47 = -23	-1

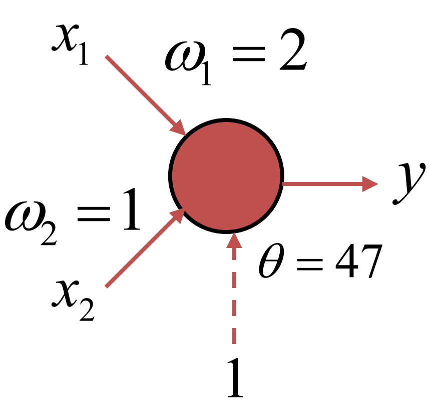
Interprétation géométrique

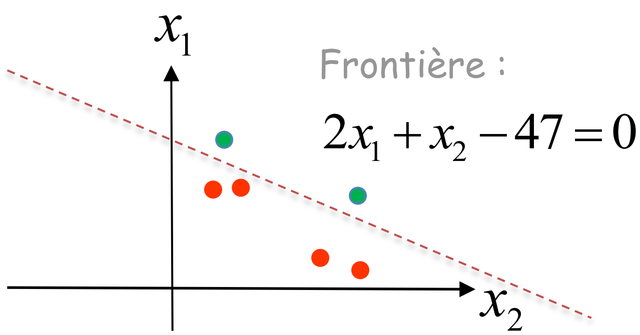




x_1	x_2	$2x_1 + x_2 - \theta$	y
20	8	48 - 47 = 1	1
15	20	50 - 47 = 3	1
16	10	42 - 47 = -5	-1
5	15	25 - 47 = -18	-1
16	6	38 - 47 = -9	-1
2	20	24 – 47 = -23	-1

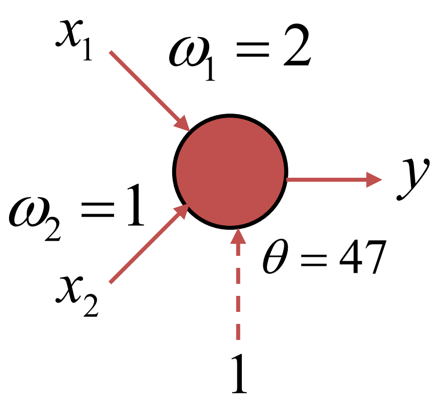
Interprétation géométrique

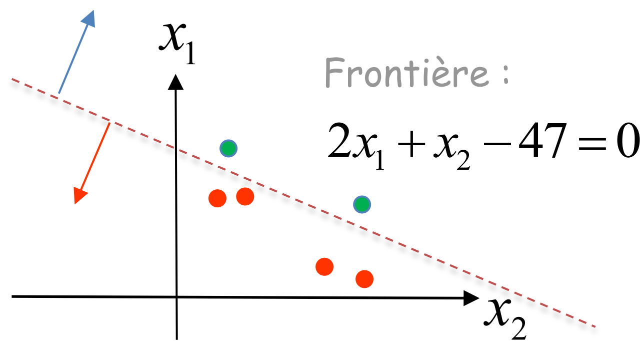




x_1	x_2	$2x_1 + x_2 - \theta$	y
20	8	48 - 47 = 1	1
15	20	50 - 47 = 3	1
16	10	42 - 47 = -5	-1
5	15	25 - 47 = -18	-1
16	6	38 - 47 = -9	-1
2	20	24 – 47 = -23	-1

Interprétation géometrique

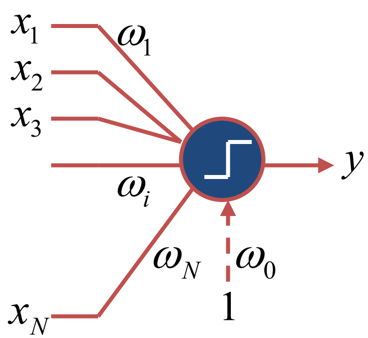




x_1	x_2	$2x_1 + x_2 - \theta$	y
20	8	48 - 47 = 1	1
15	20	50 - 47 = 3	1
16	10	42 - 47 = -5	-1
5	15	25 - 47 = -18	-1
16	6	38 - 47 = -9	-1
2	20	24 – 47 = -23	-1

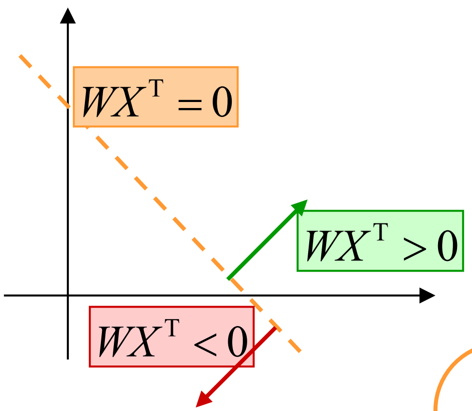
2.
Apprentissage : perspective historique

Algorithme du percetron



Formalisation vectorielle :

$$y = \mathrm{sgn}\left(\sum_{i=0}^{N} \omega_i x_i\right) = \mathrm{sgn}\left(\left[\omega_0, \omega_1, \ldots, \omega_N\right]\begin{bmatrix} x_0 \\ x_1 \\ \ldots \\ x_N \end{bmatrix}\right) = \mathrm{sgn}\left(WX^{\mathrm{T}}\right)$$



Formulation du problème :

Soient deux ensembles C_1 et C_2 de vecteurs de $\sim^{N}$

Trouver un vecteur W tel que :

$$\begin{cases} X \in C_1 \Rightarrow WX^{\mathrm{T}} > 0 \Rightarrow y = +1 \\ X \in C_2 \Rightarrow WX^{\mathrm{T}} < 0 \Rightarrow y = -1 \end{cases}$$

R. Rosenblatt, 1958.
Principles of Neurodynamics,
Spartan Books, New York 1962.

Algorithme du perceptron

Algorithme d'apprentissage du perceptron :

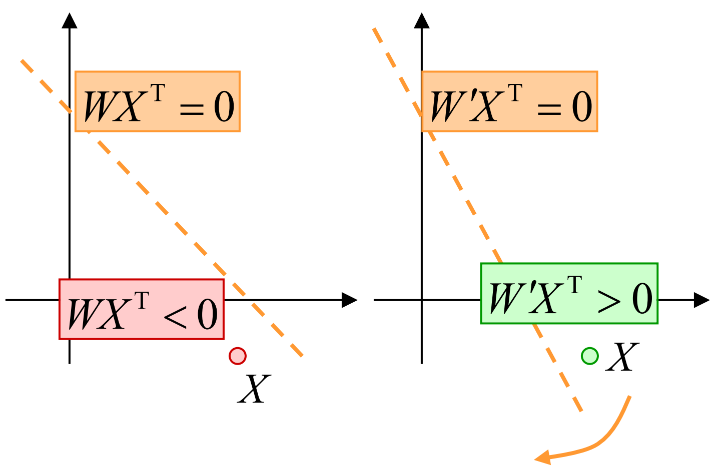
- Initialiser W aléatoirement
- Tant qu'il existe X tel que :

$$X \in C_1 \Rightarrow WX^{\mathrm{T}} > 0 \text{ et } X \in C_2 \Rightarrow WX^{\mathrm{T}} < 0 \text{ non satisfaite}$$

Faire :

$$W \leftarrow W + \lambda.\delta(X)X$$

$$\begin{cases} \lambda \text{ petite constante} \\ \begin{cases} X \in C_1 \Rightarrow \delta(X) = +1 \\ X \in C_2 \Rightarrow \delta(X) = -1 \end{cases} \end{cases}$$



$X \in C_1$ mais $WX < 0$

on cherche ΔW tel que $W'X^{\mathrm{T}} = (W + \Delta W)X^{\mathrm{T}} > 0$:

On a : $W'X^{\mathrm{T}} = (W + \lambda.1.X)X^{\mathrm{T}} = \left(WX^{\mathrm{T}} + \lambda \|X\|^2\right) > WX^{\mathrm{T}}$

$X \in C_2$ mais $WX^{\mathrm{T}} > 0$

on cherche ΔW tel que $W'X^{\mathrm{T}} = (W + \Delta W)X^{\mathrm{T}} < 0$:

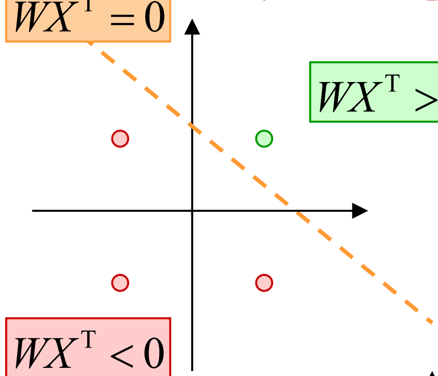
On a : $W'X^{\mathrm{T}} = (W + \lambda.(-1).X)X^{\mathrm{T}} = \left(WX - \lambda \|X\|^2\right) < WX^{\mathrm{T}}$

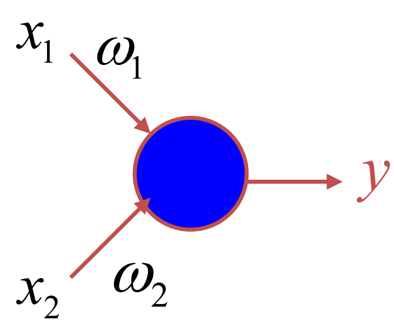
Les limites du perceptron

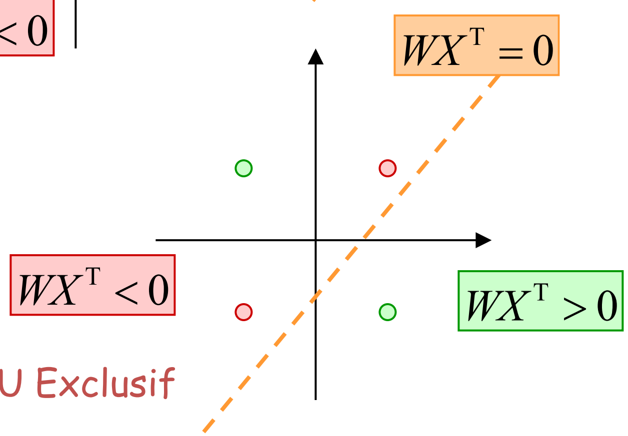
Dans une étude très détaillée, Minsky et Papert montrent en 1969 que le Perceptron ne peut s' appliquer qu'aux problèmes **linéairement séparables**

Or les problèmes de classification posés dans les applications réelles sont presque toujours **non linéairement séparables.**
Le problème non linéairement séparable le plus simple est celui du **OU Exclusif À 2 entrées**.

Fonction ET







Fonction OU Exclusif

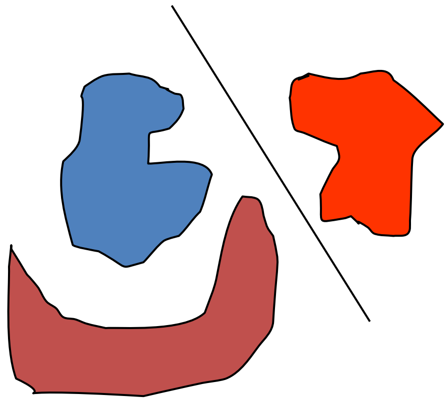
Table de vérité du OU Exclusif

Entrée 1	Entrée 2	Sortie
+1	+1	-1
+1	-1	+1
-1	+1	+1
-1	-1	-1

Les limites du perceptron

Dans une étude très détaillée, Minsky et Papert montrent en 1969 que le Perceptron ne peut s'appliquer qu'aux problèmes **linéairement séparables**

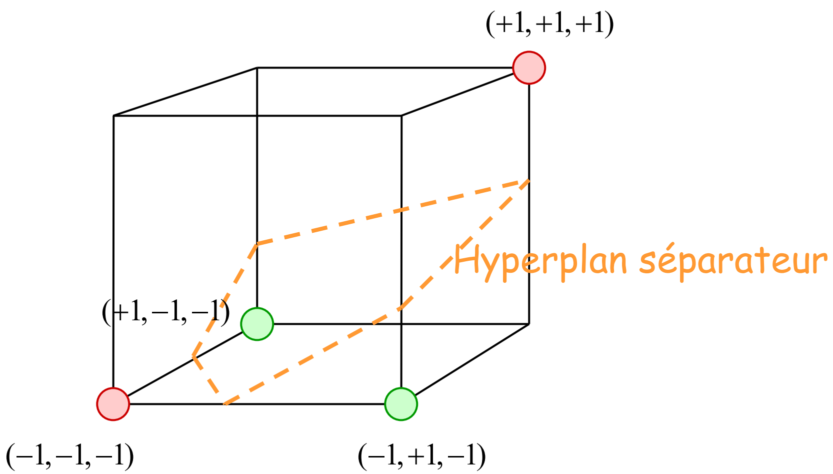
Or les problèmes de classification posés dans les applications réelles sont presque toujours **non linéairement séparables.**
Le problème non linéairement séparable le plus simple est celui du **OU Exclusif À 2 entrées**.

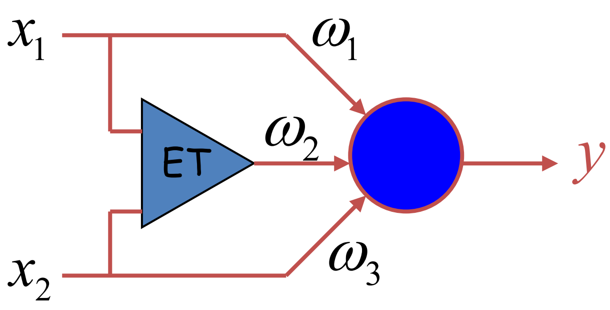


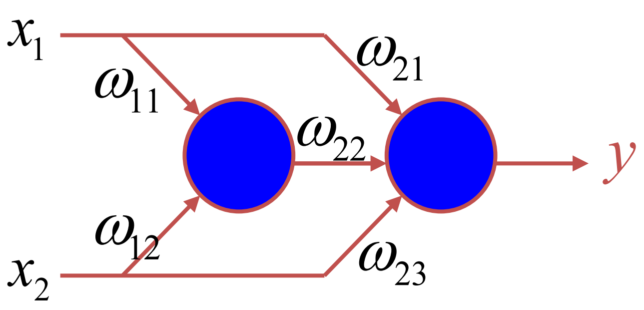
M. Minsky et S. Papert, 1969.
Perceptrons ,
the MIT Press, Cambridge 1969.

Les solutions

On peut réaliser par un perceptron une classification non linéairement séparable par changement de représentation en **augmentant la dimension.**





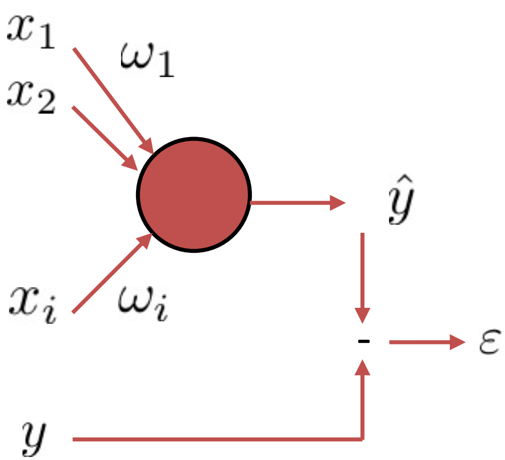


3.
L'optimisation par descente de gradient

Descente de gradient

- Etat du neurone

$$v = \sum_i \omega_i x_i$$

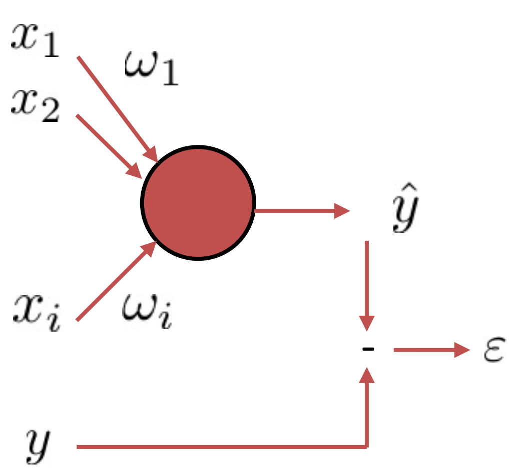


Descente de gradient

- Etat du neurone
- Sortie du neurone

$$v = \sum_i \omega_i x_i$$

$$\boxed{\hat{y} = \sigma(v)}$$



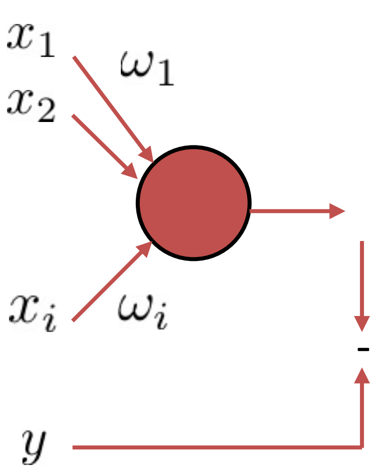
Descente de gradient

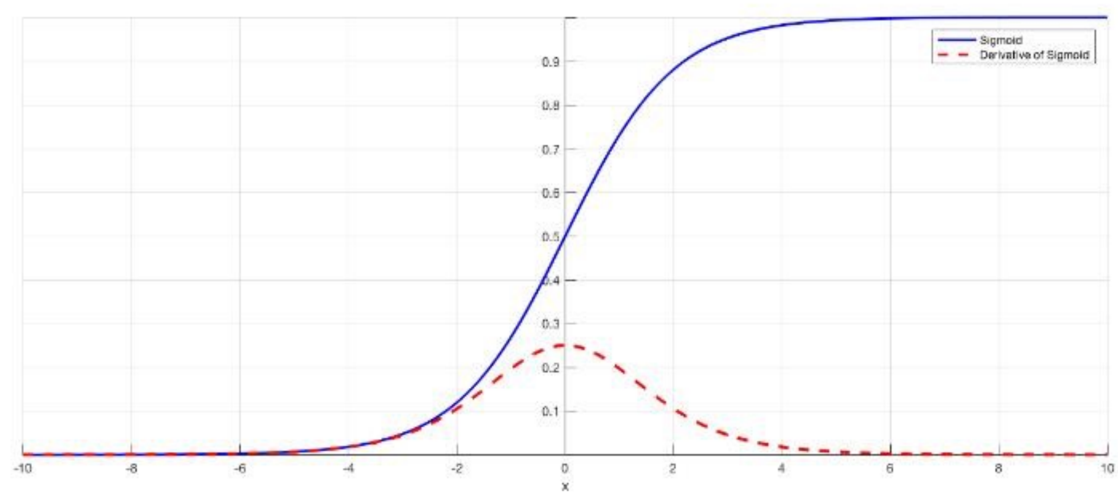
- Etat du neurone
- Sortie du neurone
- Fonction de transition

$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\boxed{\sigma(v) = \frac{1}{1 + e^{-v}}}$$





Descente de gradient

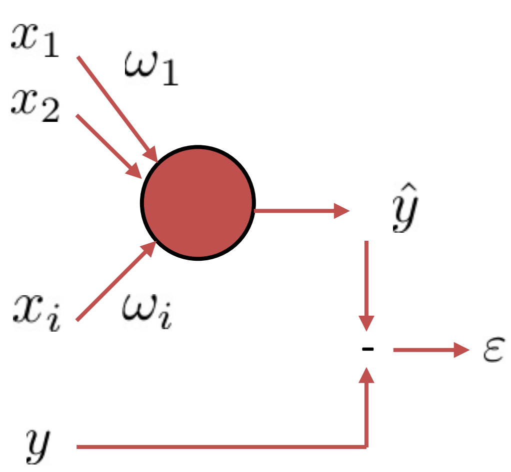
- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur

$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\boxed{\varepsilon = \hat{y} - y}$$



Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Coût (perte)

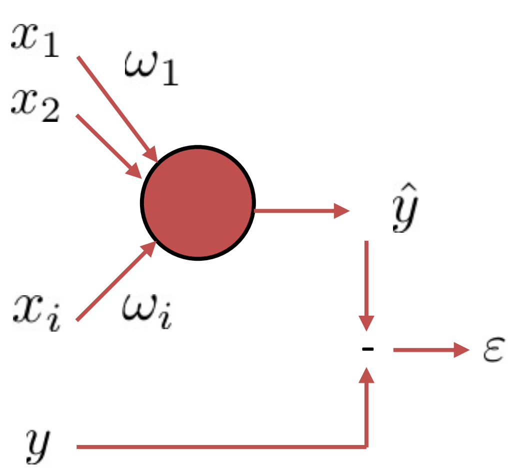
$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\boxed{\mathcal{L} = \varepsilon^2}$$



Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Coût (perte)

$$v = \sum_i \omega_i x_i$$

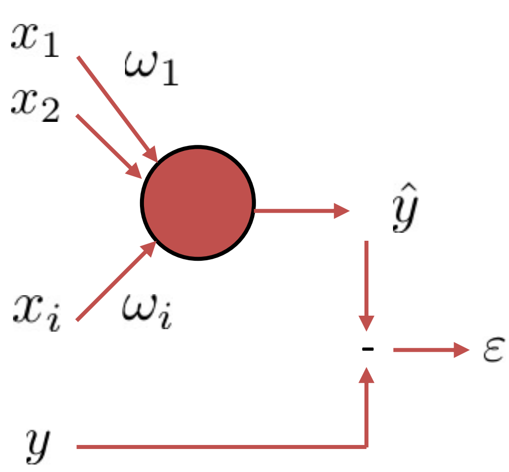
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

sur tous les exemples $\boxed{\mathcal{L} = \sum_k (\varepsilon^k)^2}$



Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique

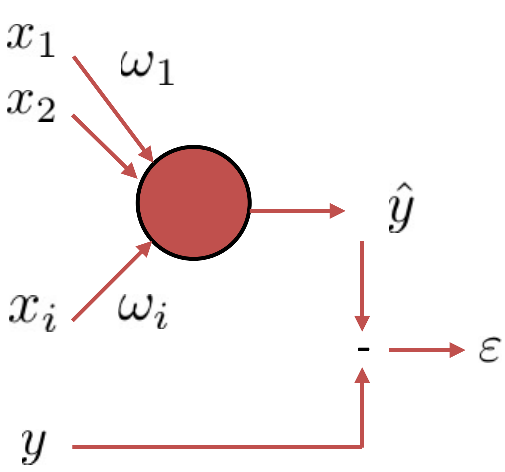
$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$



sur tous les exemples $\mathcal{L} = \sum_k (\varepsilon^k)^2$

sur toutes les sorties et tous les exemples $\boxed{\mathcal{L} = \sum_k \sum_i (\varepsilon_i^k)^2}$

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

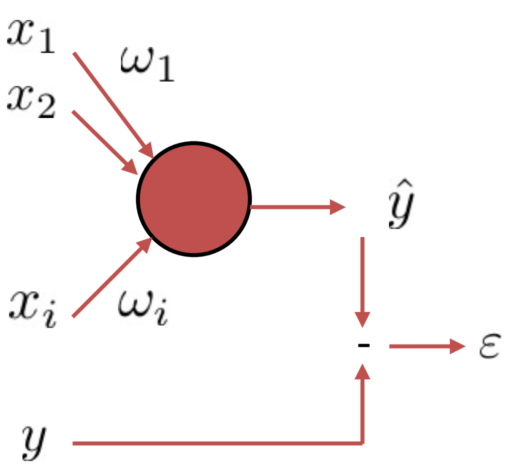
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\boxed{\omega_i \longleftarrow \omega_i + \Delta\omega_i}$$



> Modifier les poids de sorte à diminuer le coût

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

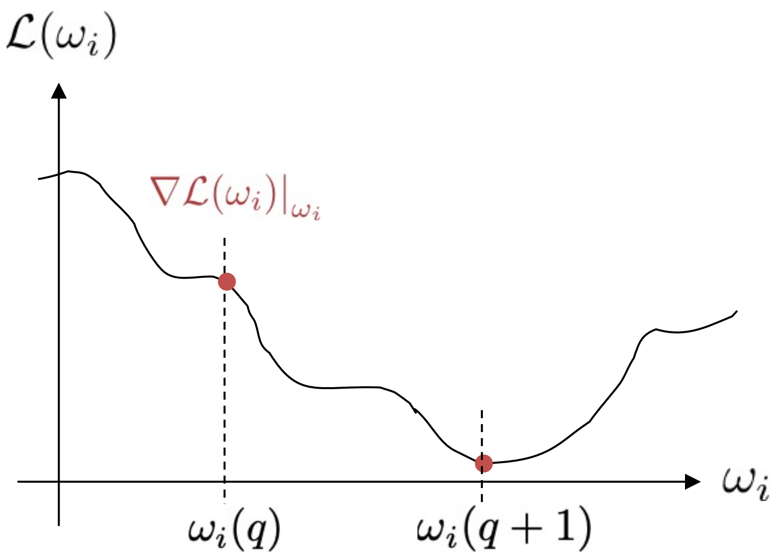
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\boxed{\omega_i \longleftarrow \omega_i + \Delta\omega_i}$$



> Allure du coût selon un paramètre du réseau

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

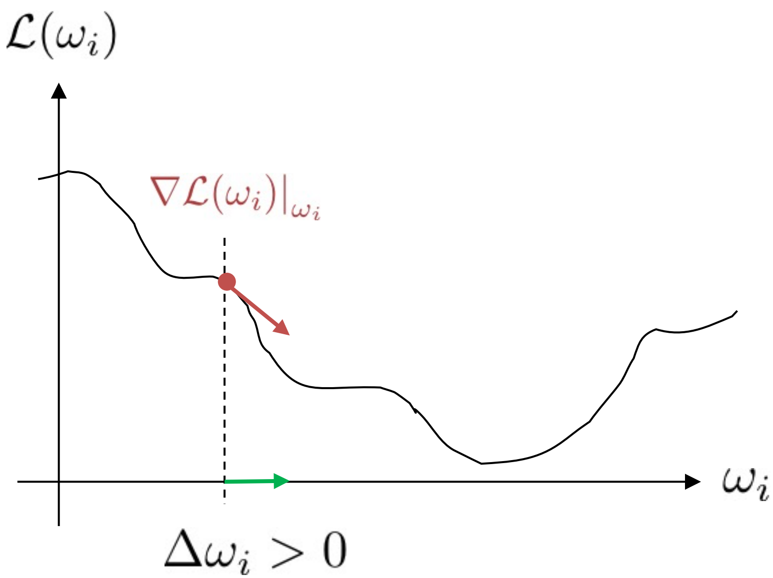
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\boxed{\omega_i \longleftarrow \omega_i + \Delta\omega_i}$$



> Approche locale: estimer la valeur du coût et sa dérivée

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

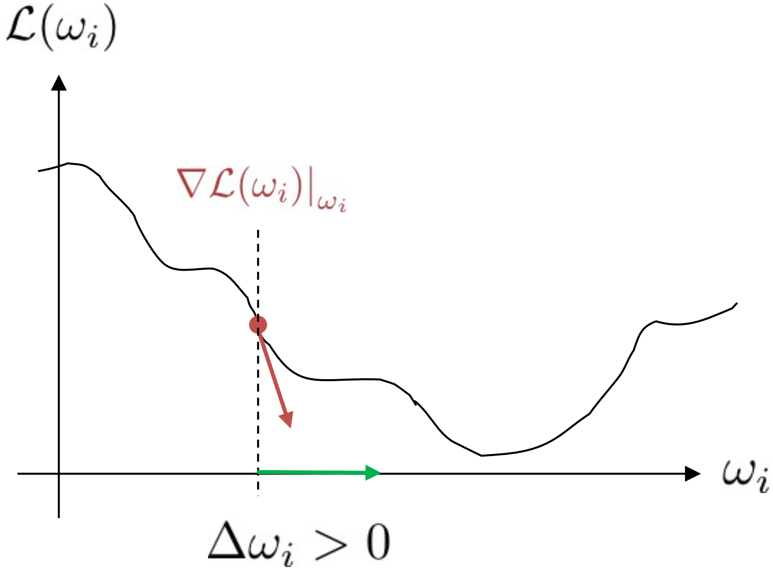
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\boxed{\omega_i \longleftarrow \omega_i + \Delta\omega_i}$$



> Approche locale: estimer la valeur du coût et sa dérivée

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

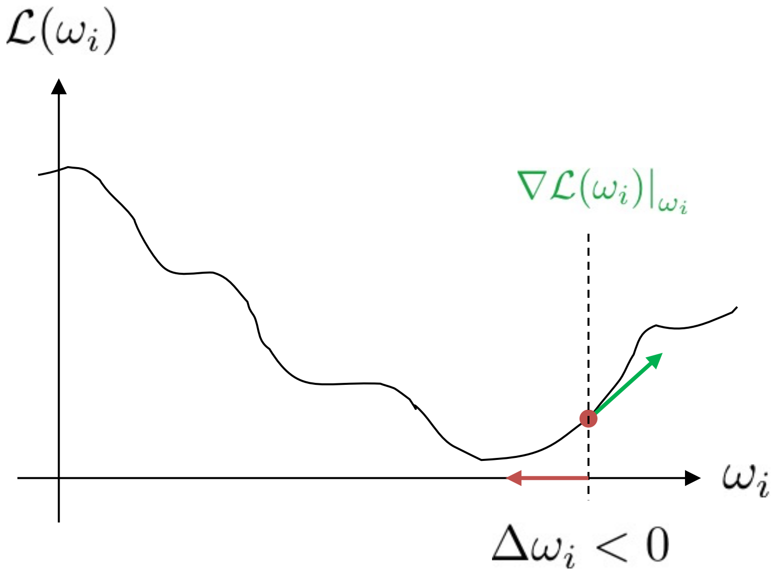
$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

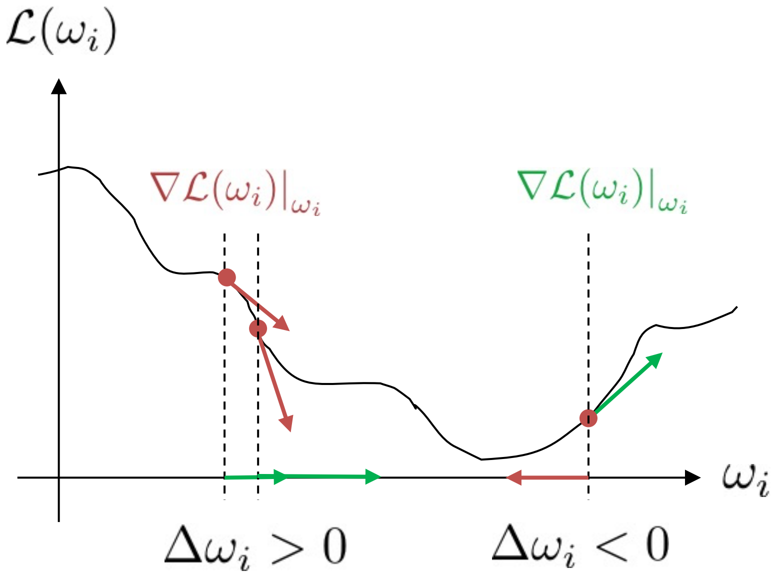
$$\boxed{\omega_i \longleftarrow \omega_i + \Delta\omega_i}$$



Descente de gradient

- Etat du neurone $$v = \sum_i \omega_i x_i$$
- Sortie du neurone $$\hat{y} = \sigma(v)$$
- Fonction de transition $$\sigma(v) = \frac{1}{1+e^{-v}}$$
- Erreur $$\varepsilon = \hat{y} - y$$
- Erreur quadratique $$\mathcal{L} = \varepsilon^2$$
- Adaptation $$\omega_i \longleftarrow \omega_i + \Delta\omega_i$$

$$\boxed{\Delta\omega_i = -\nabla\mathcal{L}|_{\omega_i}}$$



> Approche locale: modifier dans le sens opposé au gradient

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Adaptation

$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

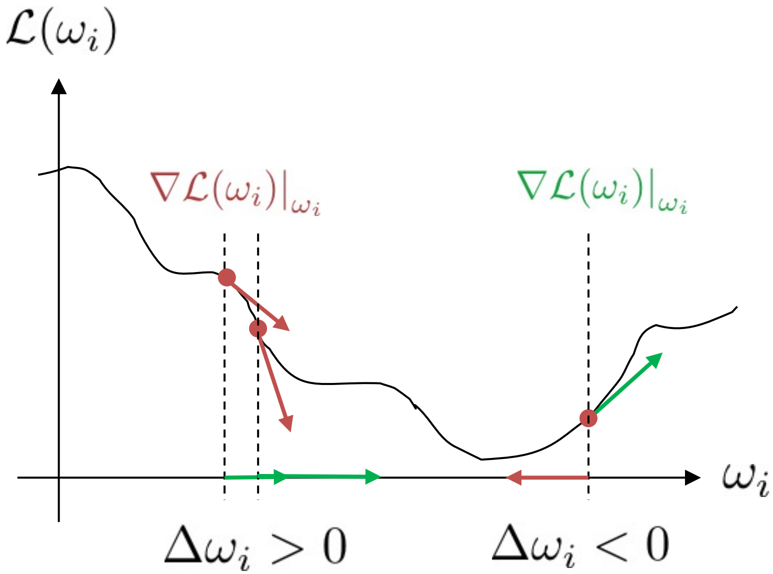
$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\omega_i \longleftarrow \omega_i + \Delta\omega_i$$

$$\Delta\omega_i = -\nabla\mathcal{L}|_{\omega_i}$$

$$\boxed{\omega_i \longleftarrow \omega_i - \lambda\nabla\mathcal{L}|_{\omega_i}}$$



> Attention au pas d'adaptation

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Gradient
- Adaptation

$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\frac{\partial \mathcal{L}(\omega_i)}{\partial \omega_i} = 2(\hat{y} - y)\frac{\partial \sigma(\omega_i)}{\partial \omega_i} = 2(\hat{y} - y)\sigma'(\omega_i)x_i$$

$$\boxed{\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}}$$

Descente de gradient

- Etat du neurone
- Sortie du neurone
- Fonction de transition
- Erreur
- Erreur quadratique
- Gradient
- Adaptation

$$v = \sum_i \omega_i x_i$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\frac{\partial \mathcal{L}(\omega_i)}{\partial \omega_i} = 2(\hat{y} - y)\frac{\partial \sigma(\omega_i)}{\partial \omega_i} = 2(\hat{y} - y)\sigma'(\omega_i)x_i$$

$$\boxed{\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i}$$

$$\sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

SCIENCES SORBONNE UNIVERSITÉ

4.
Calcul du gradient

Calcul du gradient

- Etat des entrées

$$\boxed{\begin{gathered} e_i = \omega_i x_i \\ v = \sum_i e_i + \theta \end{gathered}}$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

> Calcul: introduire des variables intermédiaires

Calcul du gradient

- Dérivées des fonctions composées

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i}\boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$\boxed{e_i = \omega_i x_i}$$

$$v = \sum_i e_i + \theta$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

> Calcul: décomposition des dérivées selon les variables intermédiaires

Calcul du gradient

- Dérivées des fonctions composées

$$e_i = \omega_i x_i$$

$$v = \sum_i e_i + \theta$$

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i} \boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial v} \boxed{\frac{\partial v}{\partial e_i}}} \longrightarrow 1$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1 + e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

Calcul du gradient

- Dérivées des fonctions composées

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i}\boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial v}\boxed{\frac{\partial v}{\partial e_i}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}\boxed{\frac{\partial \hat{y}}{\partial v}}} \longrightarrow \sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

$$\boxed{e_i = \omega_i x_i}$$

$$\boxed{v = \sum_i e_i + \theta}$$

$$\boxed{\hat{y} = \sigma(v)}$$

$$\boxed{\sigma(v) = \frac{1}{1+e^{-v}}}$$

$$\varepsilon = \hat{y} - y$$

$$\mathcal{L} = \varepsilon^2$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

Calcul du gradient

- Dérivées des fonctions composées

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i}\boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial v}\boxed{\frac{\partial v}{\partial e_i}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}\boxed{\frac{\partial \hat{y}}{\partial v}}} \longrightarrow \sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \varepsilon}\boxed{\frac{\partial \varepsilon}{\partial \hat{y}}}} \longrightarrow 1$$

$$\boxed{e_i = \omega_i x_i}$$

$$\boxed{v = \sum_i e_i + \theta}$$

$$\boxed{\hat{y} = \sigma(v)}$$

$$\boxed{\sigma(v) = \frac{1}{1+e^{-v}}}$$

$$\boxed{\varepsilon = \hat{y} - y}$$

$$\mathcal{L} = \varepsilon^2$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

Calcul du gradient

- Dérivées des fonctions composées

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i}\boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial v}\boxed{\frac{\partial v}{\partial e_i}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}\boxed{\frac{\partial \hat{y}}{\partial v}}} \longrightarrow \sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \varepsilon}\boxed{\frac{\partial \varepsilon}{\partial \hat{y}}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \mathcal{L}}\boxed{\frac{\partial \mathcal{L}}{\partial \varepsilon}}} \longrightarrow 2\varepsilon$$

$$1$$

$$\boxed{e_i = \omega_i x_i}$$

$$\boxed{v = \sum_i e_i + \theta}$$

$$\boxed{\hat{y} = \sigma(v)}$$

$$\boxed{\sigma(v) = \frac{1}{1+e^{-v}}}$$

$$\boxed{\varepsilon = \hat{y} - y}$$

$$\boxed{\mathcal{L} = \varepsilon^2}$$

$$\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

Calcul du gradient

- Dérivées des fonctions composées

$$\frac{\partial \mathcal{L}}{\partial \omega_i} = \frac{\partial \mathcal{L}}{\partial e_i}\boxed{\frac{\partial e_i}{\partial \omega_i}} \longrightarrow x_i$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial v}\boxed{\frac{\partial v}{\partial e_i}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \hat{y}}\boxed{\frac{\partial \hat{y}}{\partial v}}} \longrightarrow \sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \varepsilon}\boxed{\frac{\partial \varepsilon}{\partial \hat{y}}}} \longrightarrow 1$$

$$= \boxed{\frac{\partial \mathcal{L}}{\partial \mathcal{L}}\boxed{\frac{\partial \mathcal{L}}{\partial \varepsilon}}} \longrightarrow 2\varepsilon$$

$$\downarrow$$

$$1$$

$$e_i = \omega_i x_i$$

$$v = \sum_i e_i + \theta$$

$$\hat{y} = \sigma(v)$$

$$\sigma(v) = \frac{1}{1+e^{-v}}$$

$$\varepsilon = \hat{y} - y$$

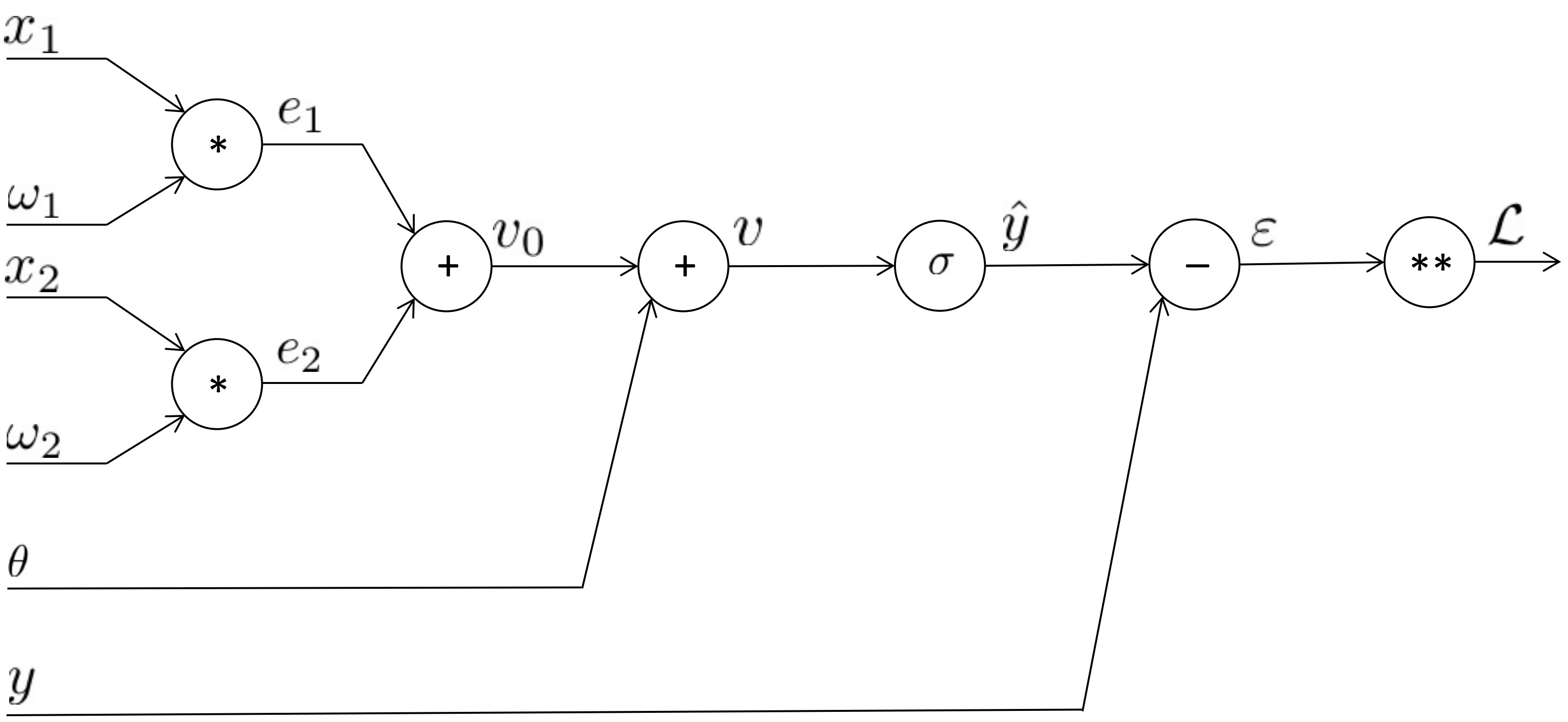
$$\mathcal{L} = \varepsilon^2$$

$$\boxed{\nabla \mathcal{L}|_{\omega_i} = 2(\varepsilon)\sigma'(v)x_i}$$

$$\omega_i \longleftarrow \omega_i - \lambda \nabla \mathcal{L}|_{\omega_i}$$

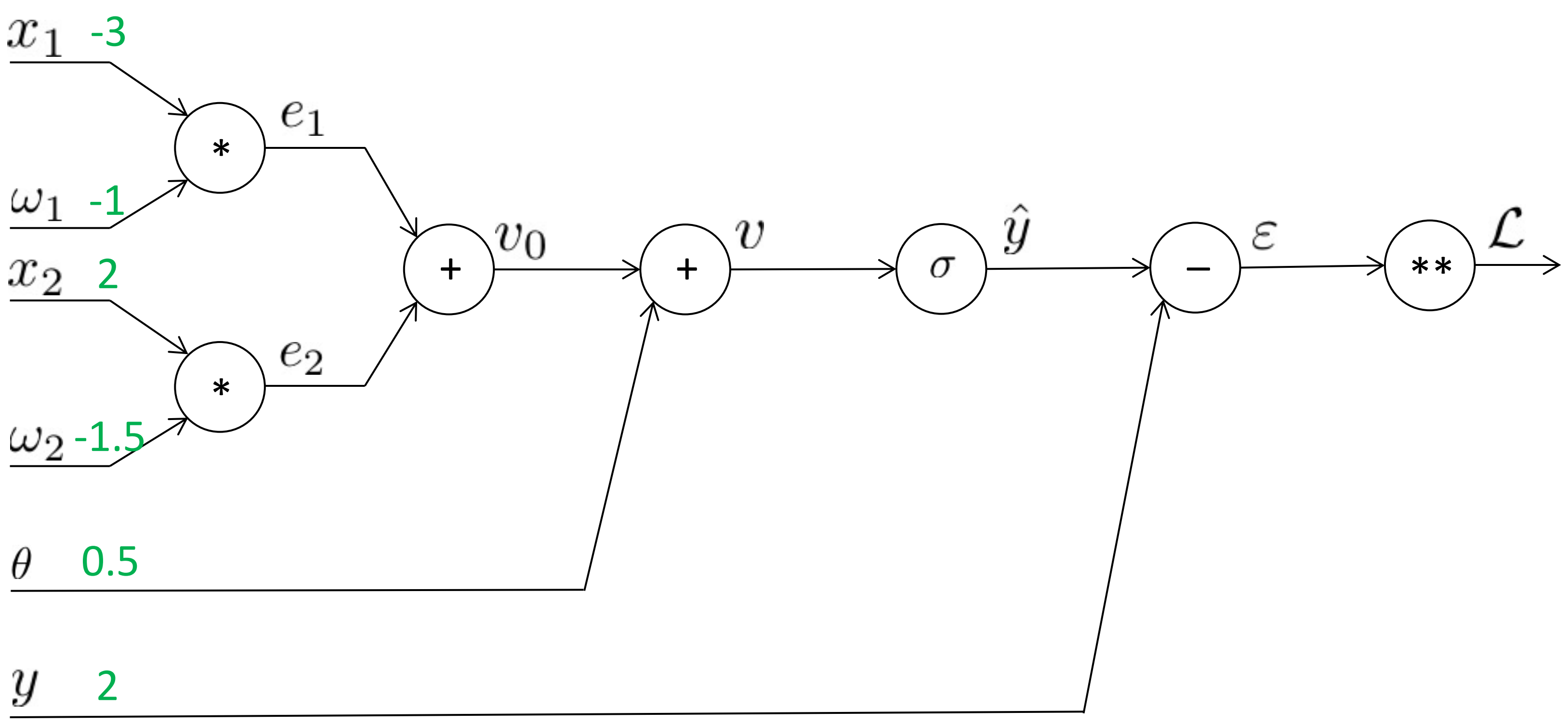
Calcul du gradient

- Propagation: construction du graphe de calccul



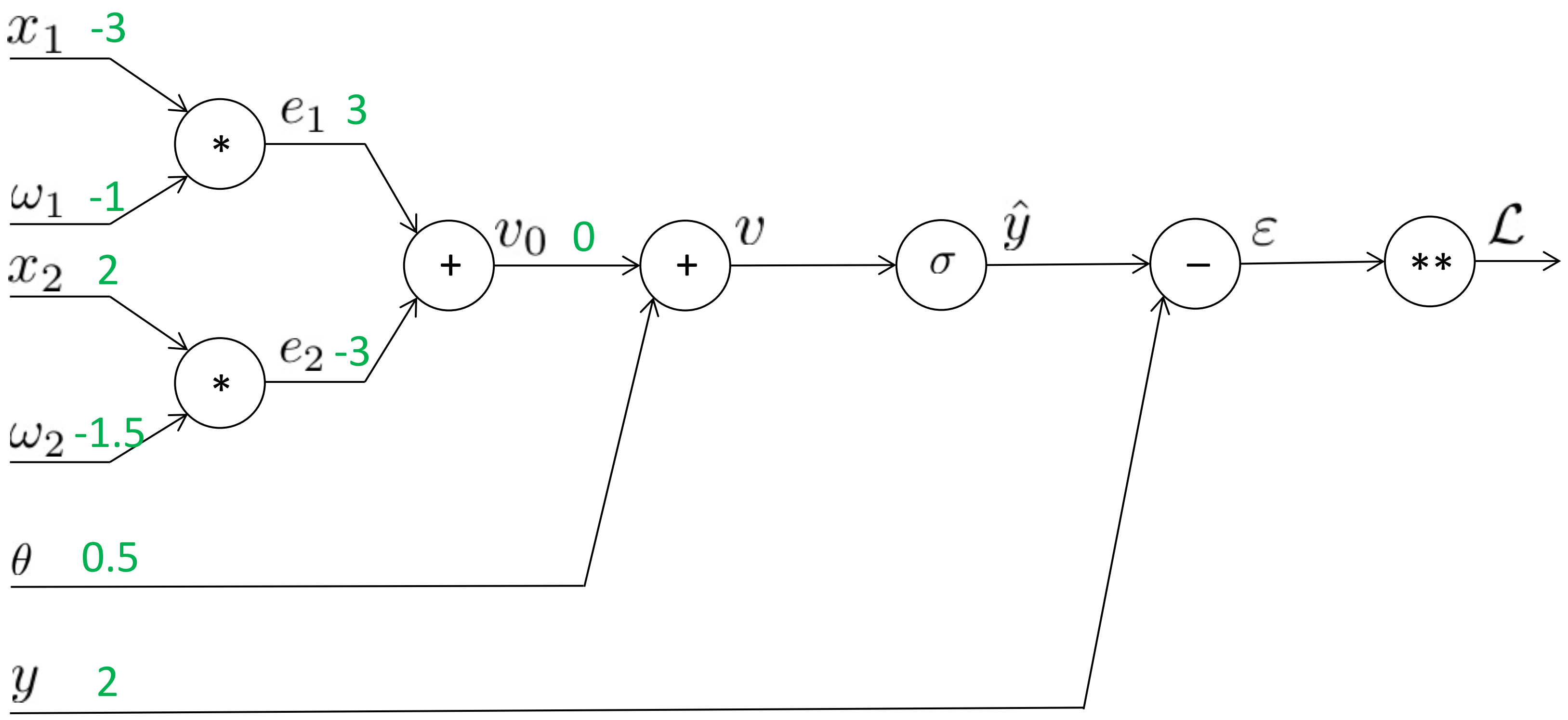
Calcul du gradient

- Propagation d'une activité



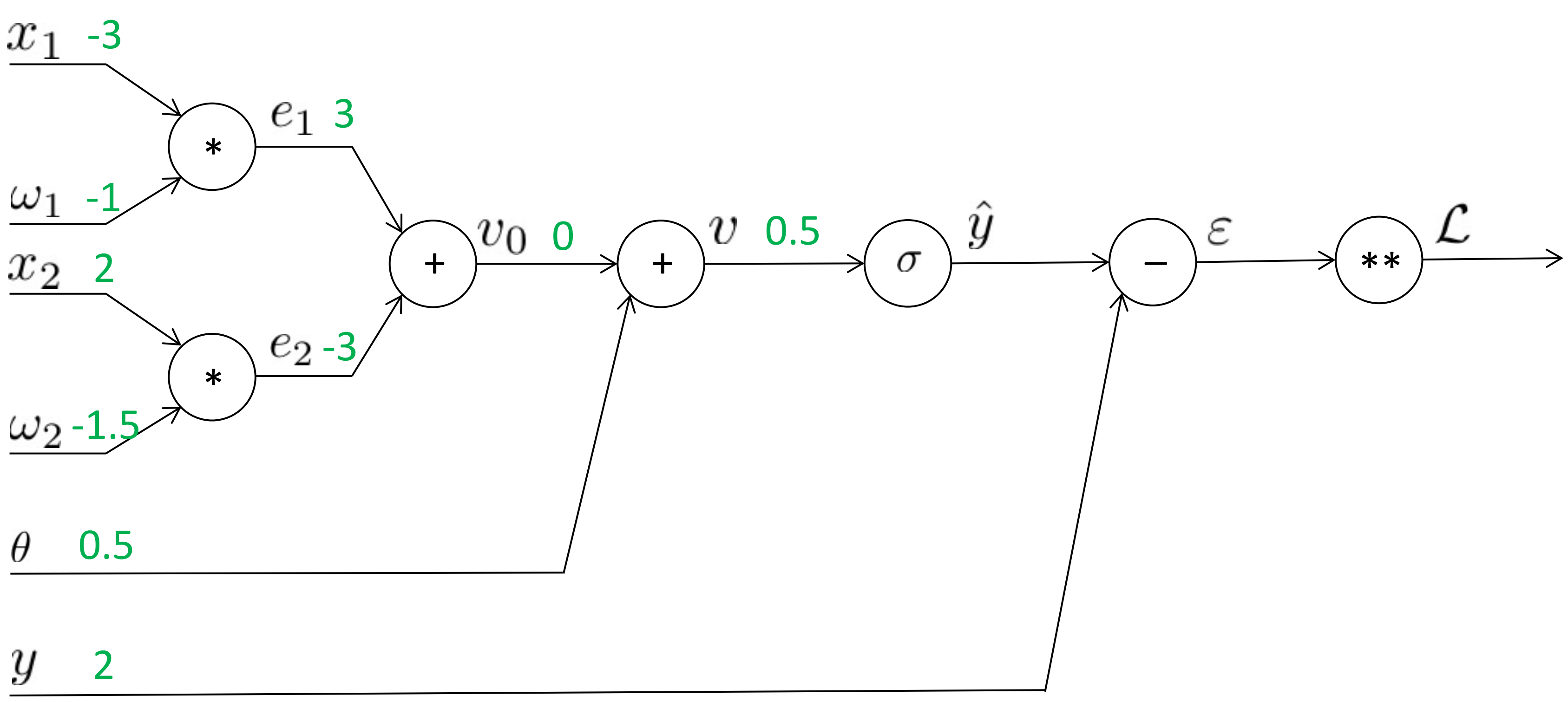
Calcul du gradient

- Propagation



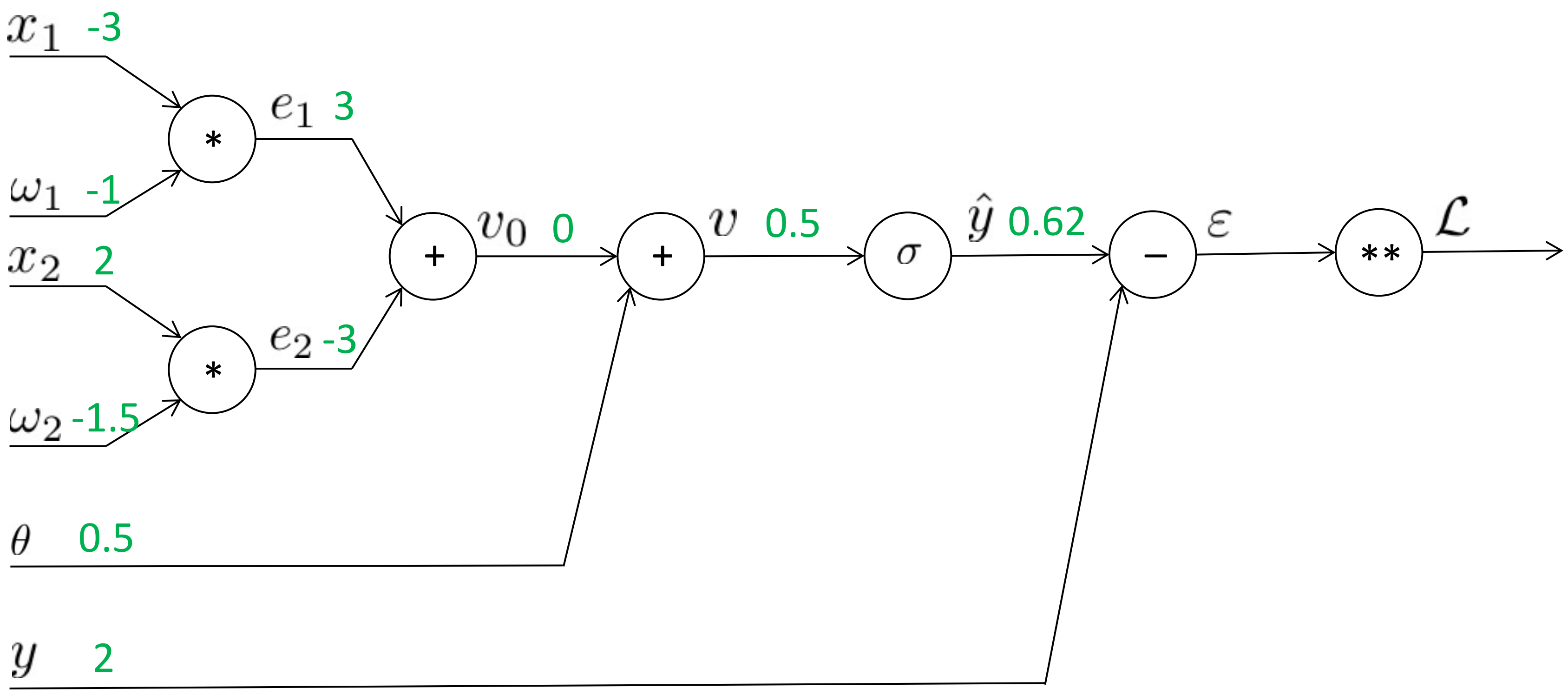
Calcul du gradient

- Propagation



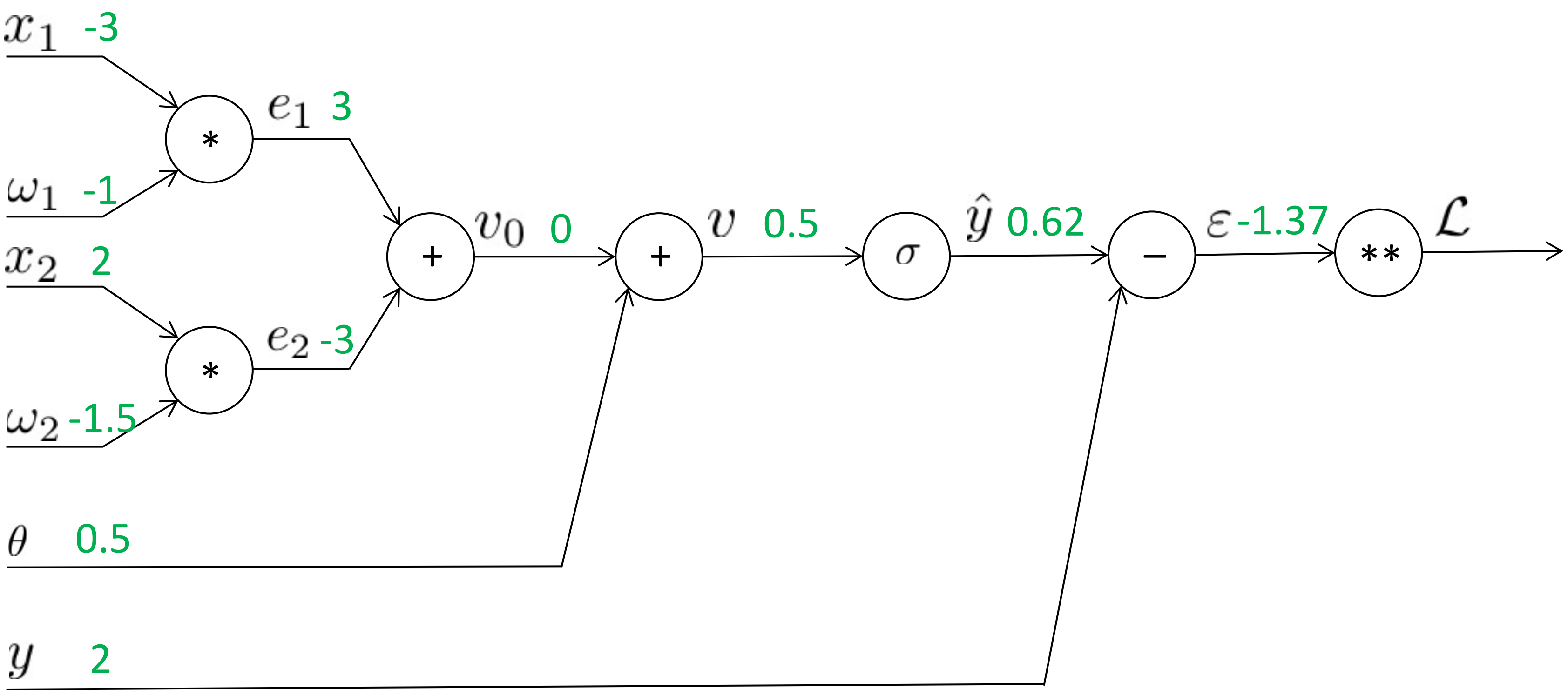
Calcul du gradient

- Propagation



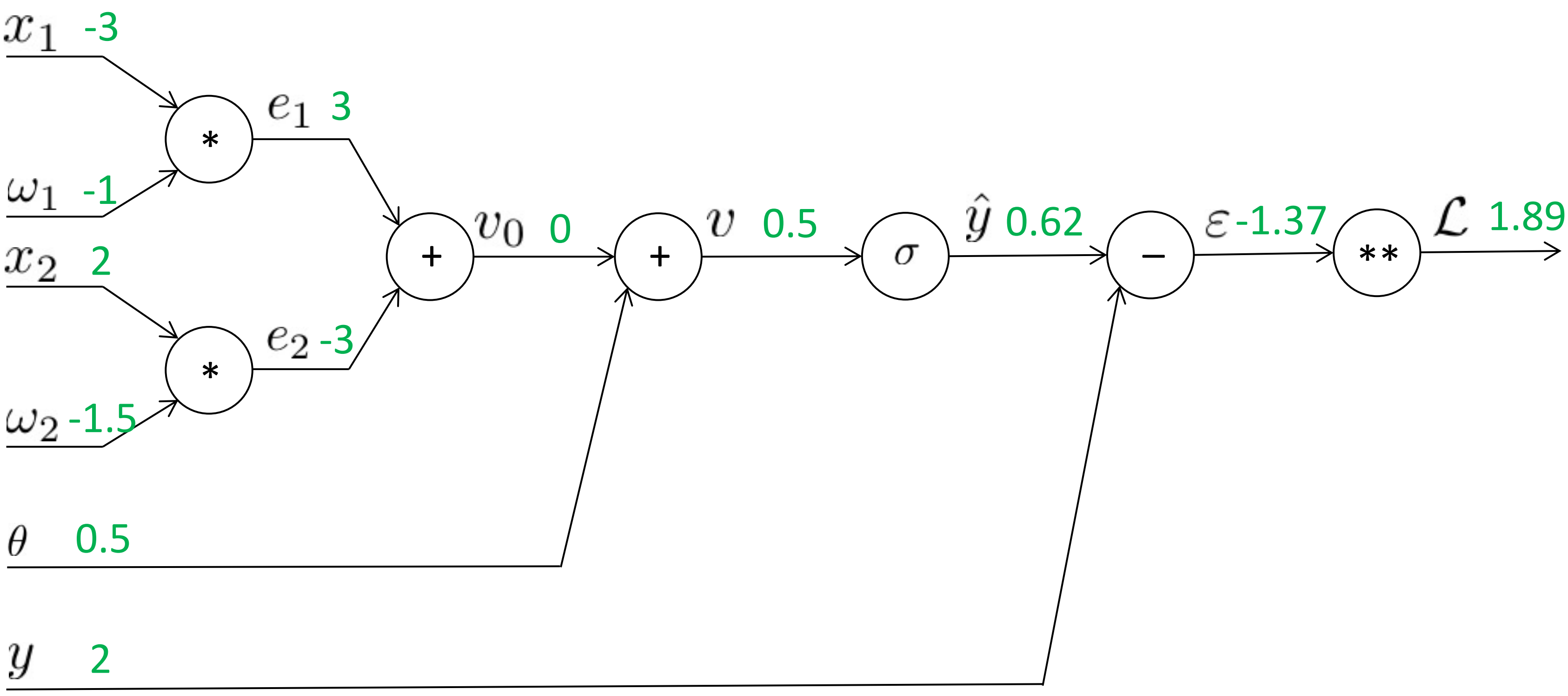
Calcul du gradient

- Propagation



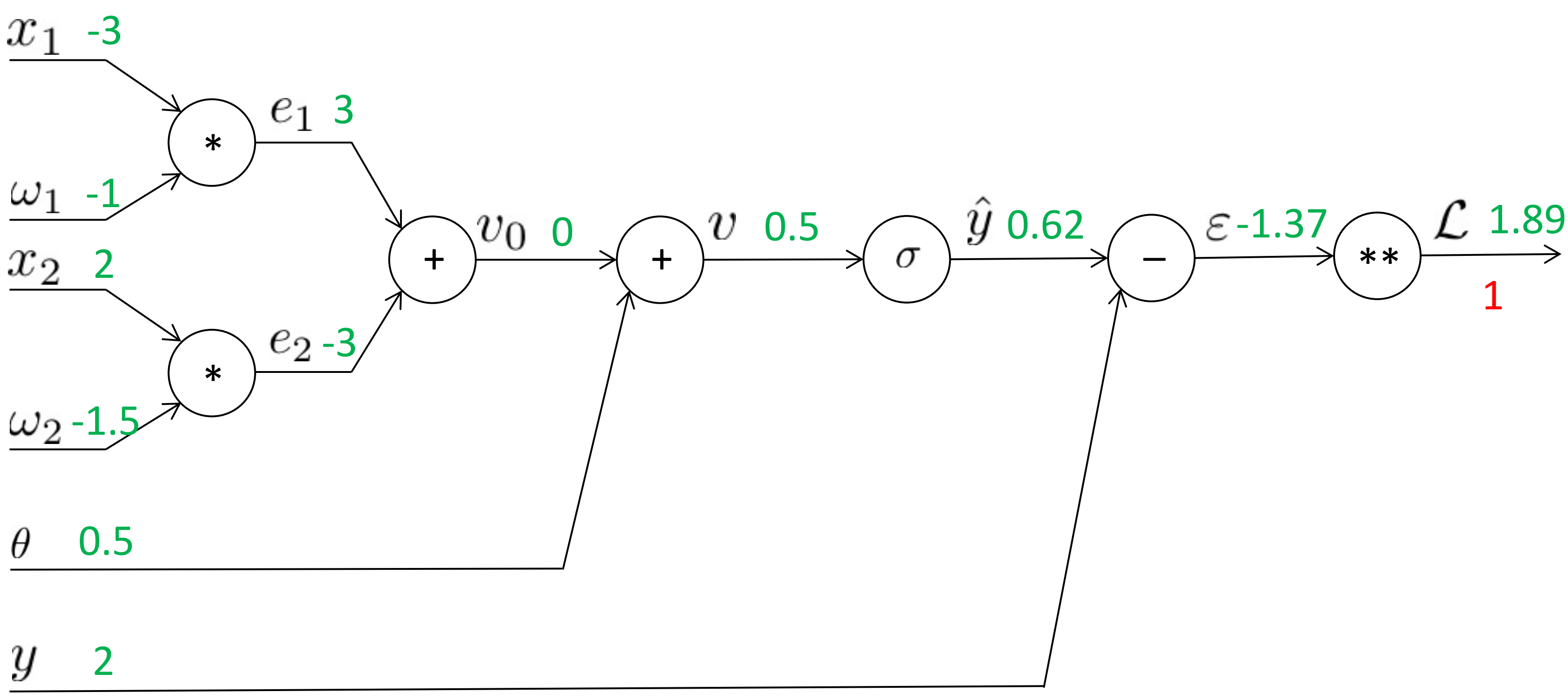
Calcul du gradient

- Propagation



Calcul du gradient

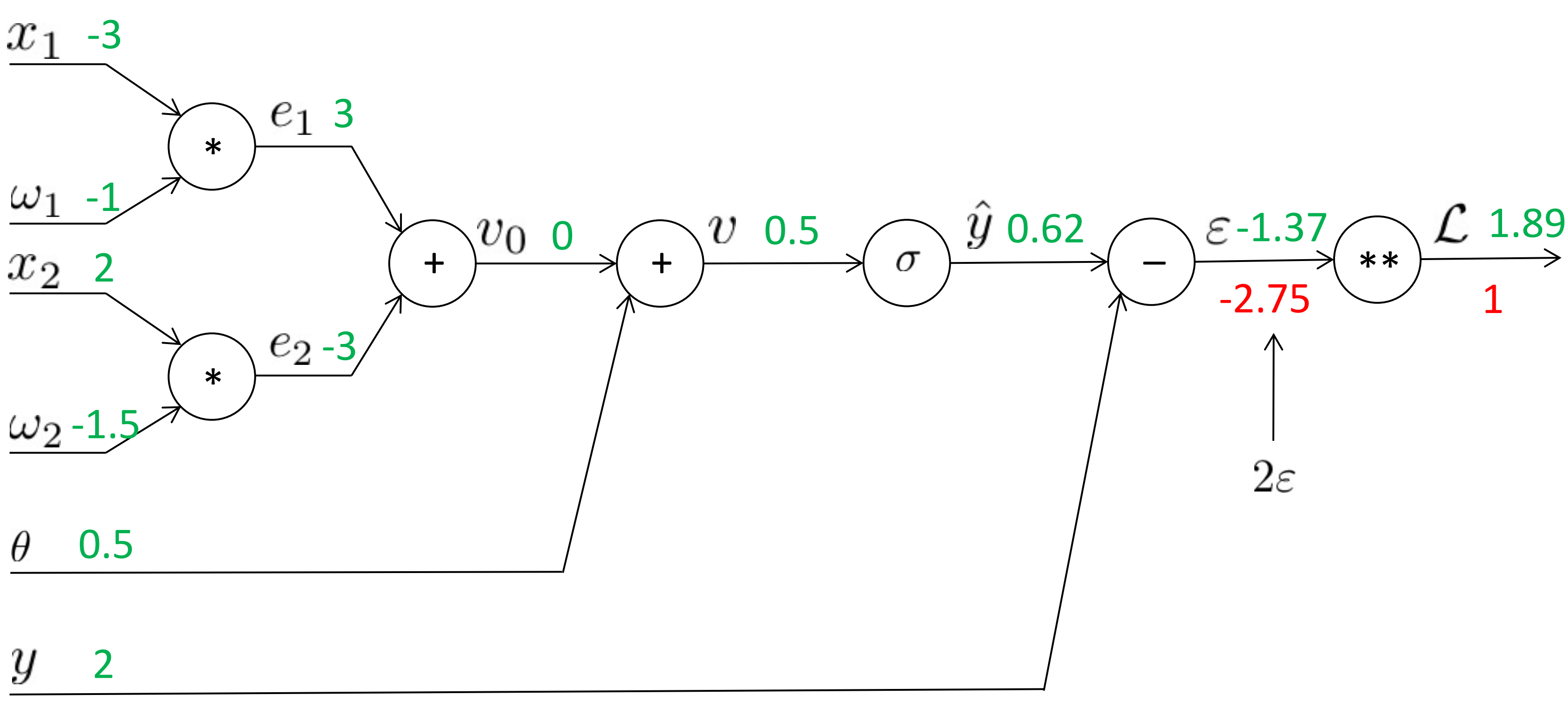
- Rétro-propagation



> Gradient du coût par rapport à lui-même: 1

Calcul du gradient

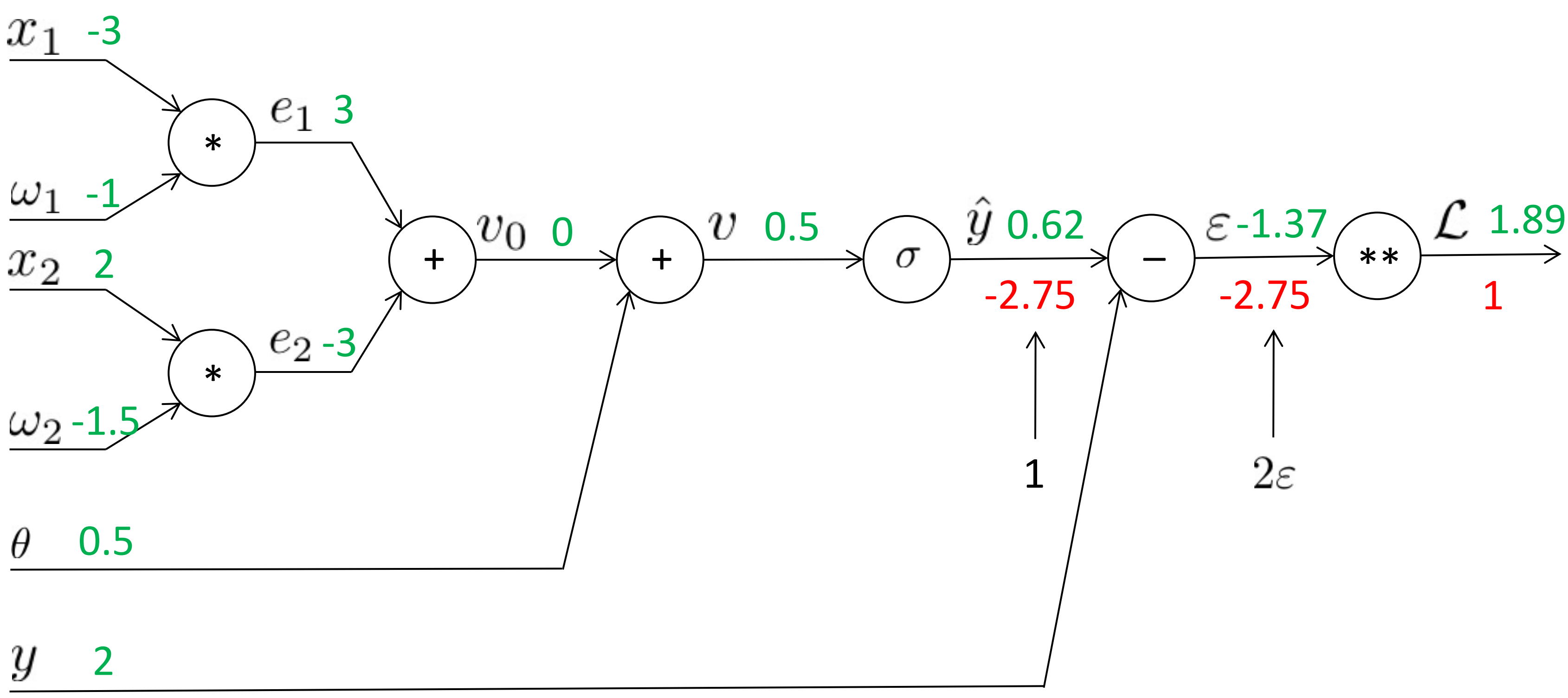
- Rétro-propagation



➢ L'erreur est déjà calculée ! Le gradient est la dérivée locale multipliée par le gradient rétropropagé

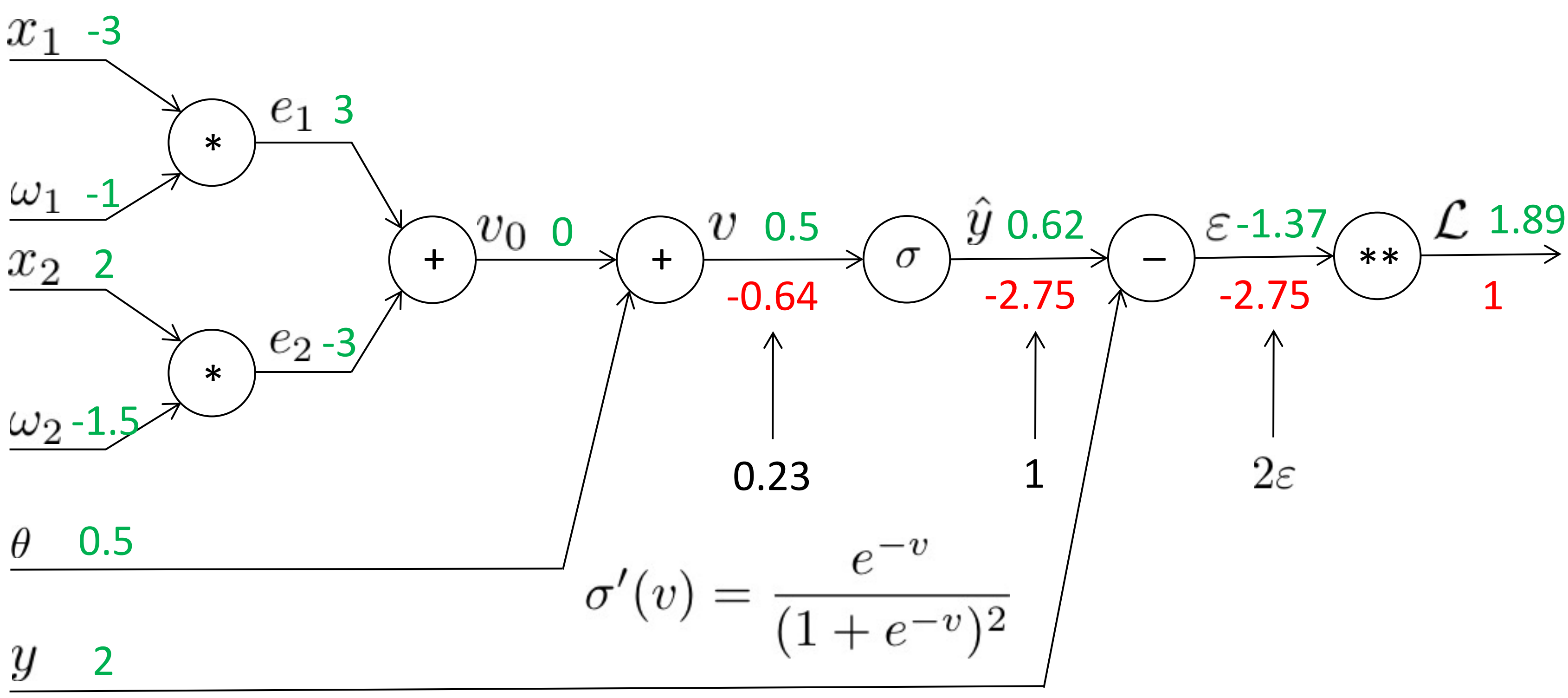
Calcul du gradient

- Rétro-propagation



Calcul du gradient

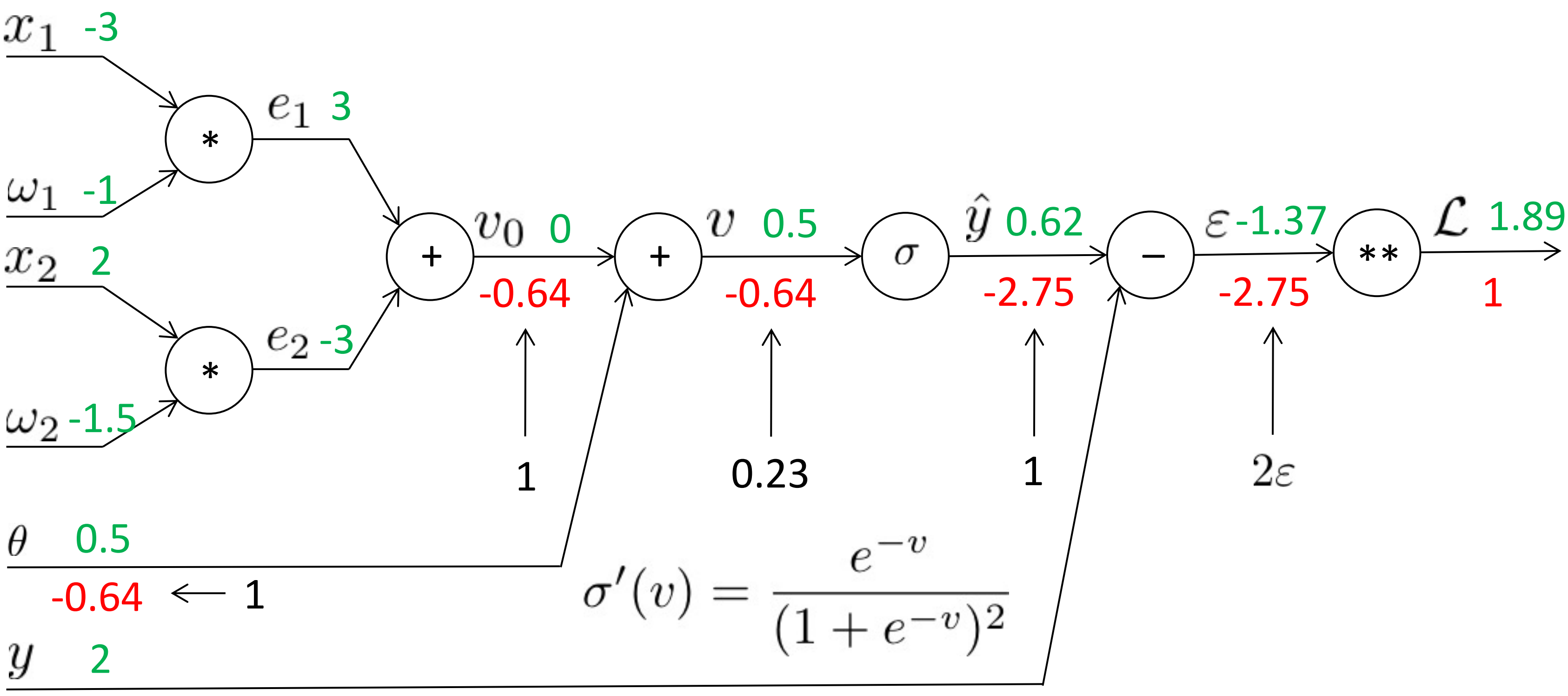
- Rétro-propagation



> Le potentiel est déjà calculé !

Calcul du gradient

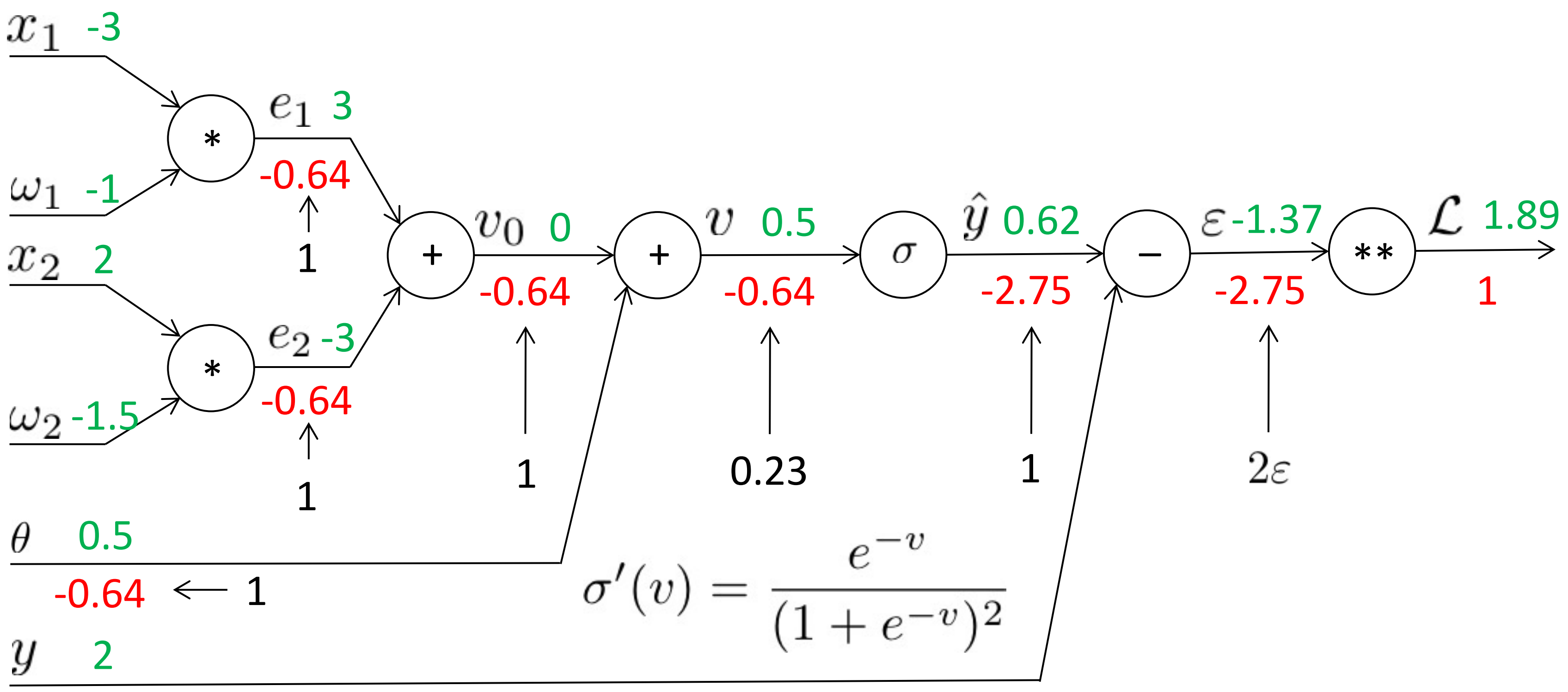
- Rétro-propagation



$$\sigma'(v) = \frac{e^{-v}}{(1+e^{-v})^2}$$

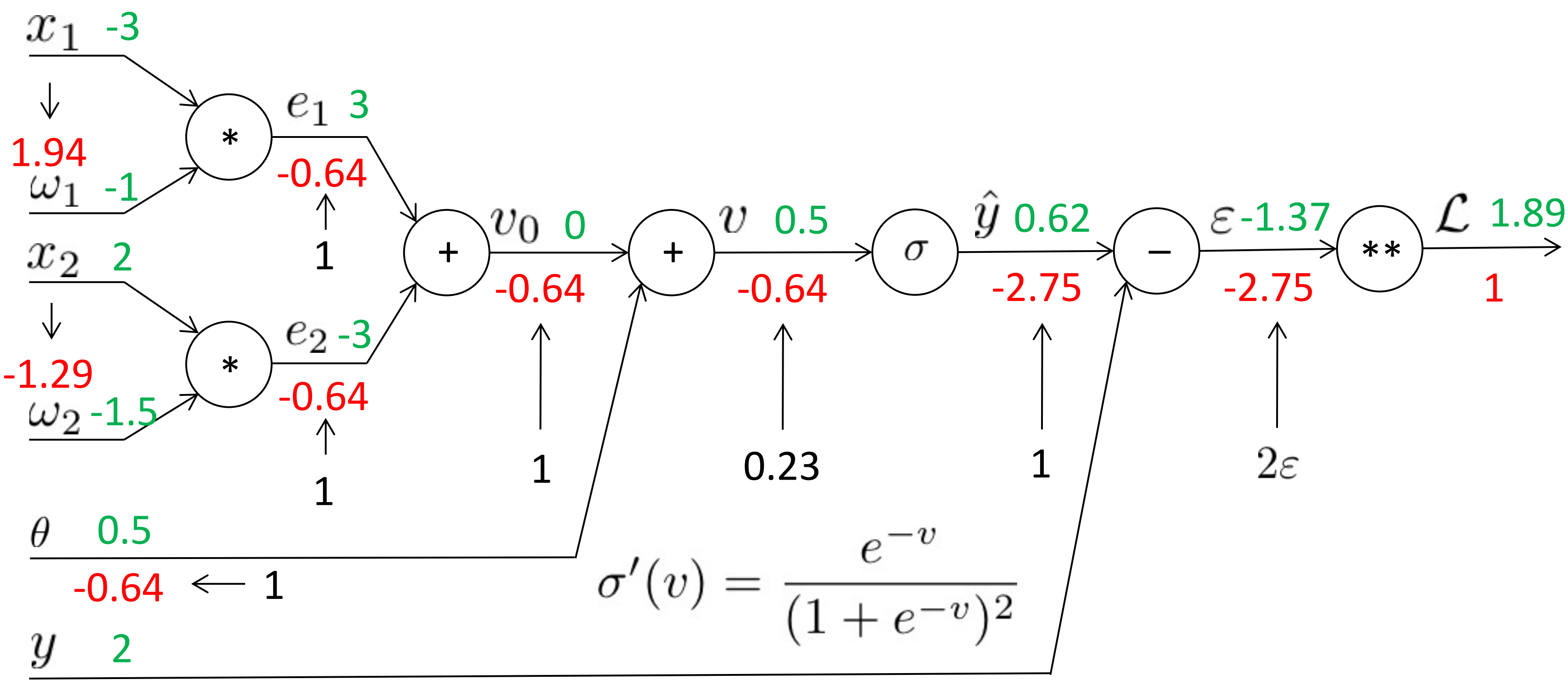
Calcul du gradient

- Rétro-propagation



Calcul du gradient

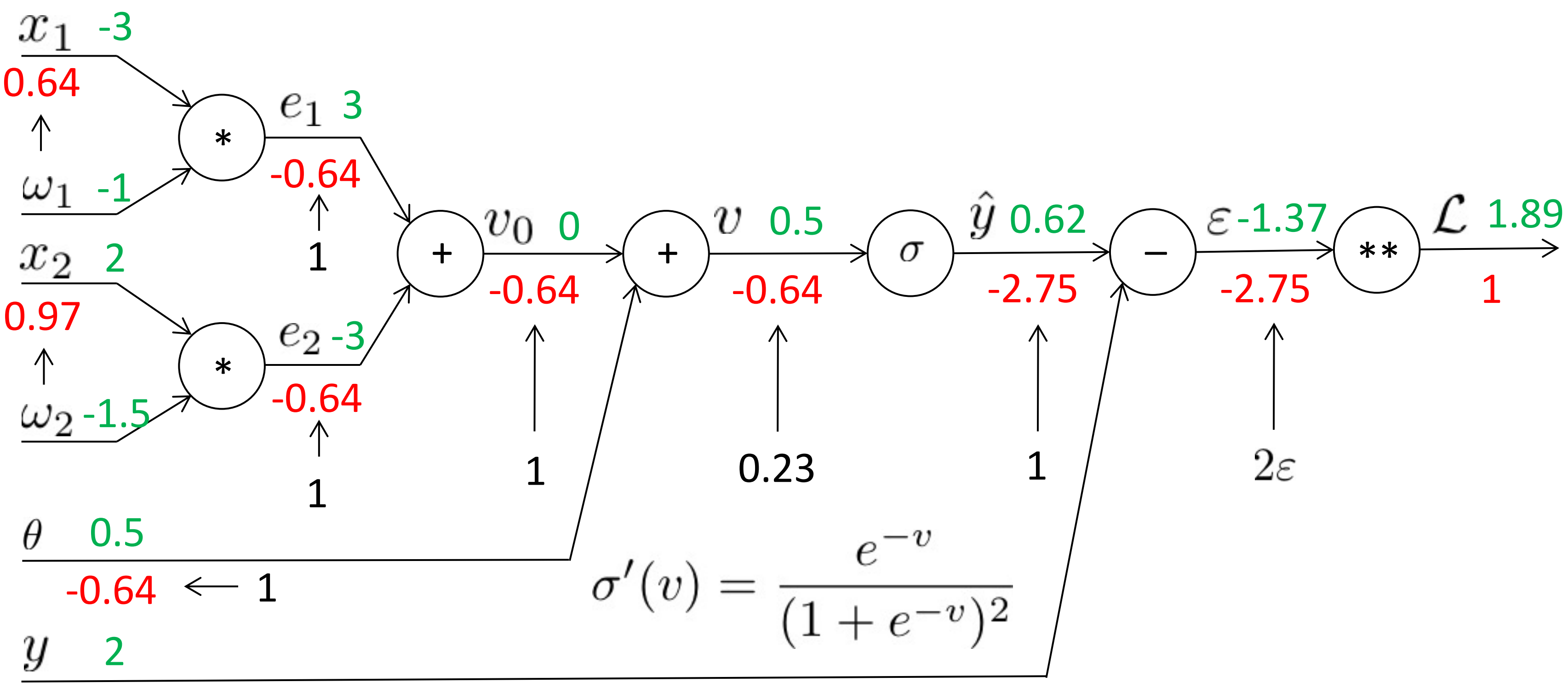
- Rétro-propagation



> Le calcul est terminé (vraiment ?)

Calcul du gradient

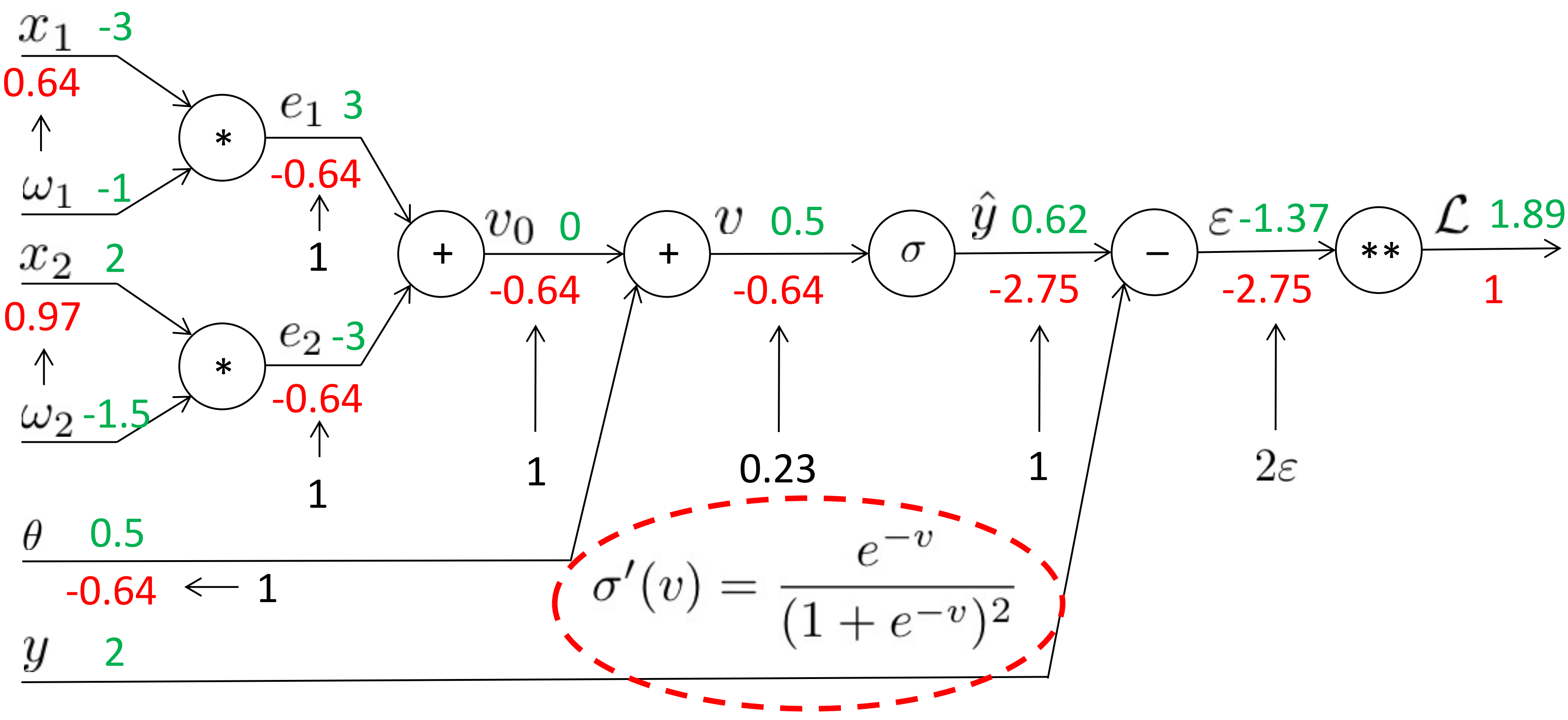
- Rétro-propagation



> Gradient relativement aux entrées: pour continuer sur des couches précédentes

Calcul du gradient

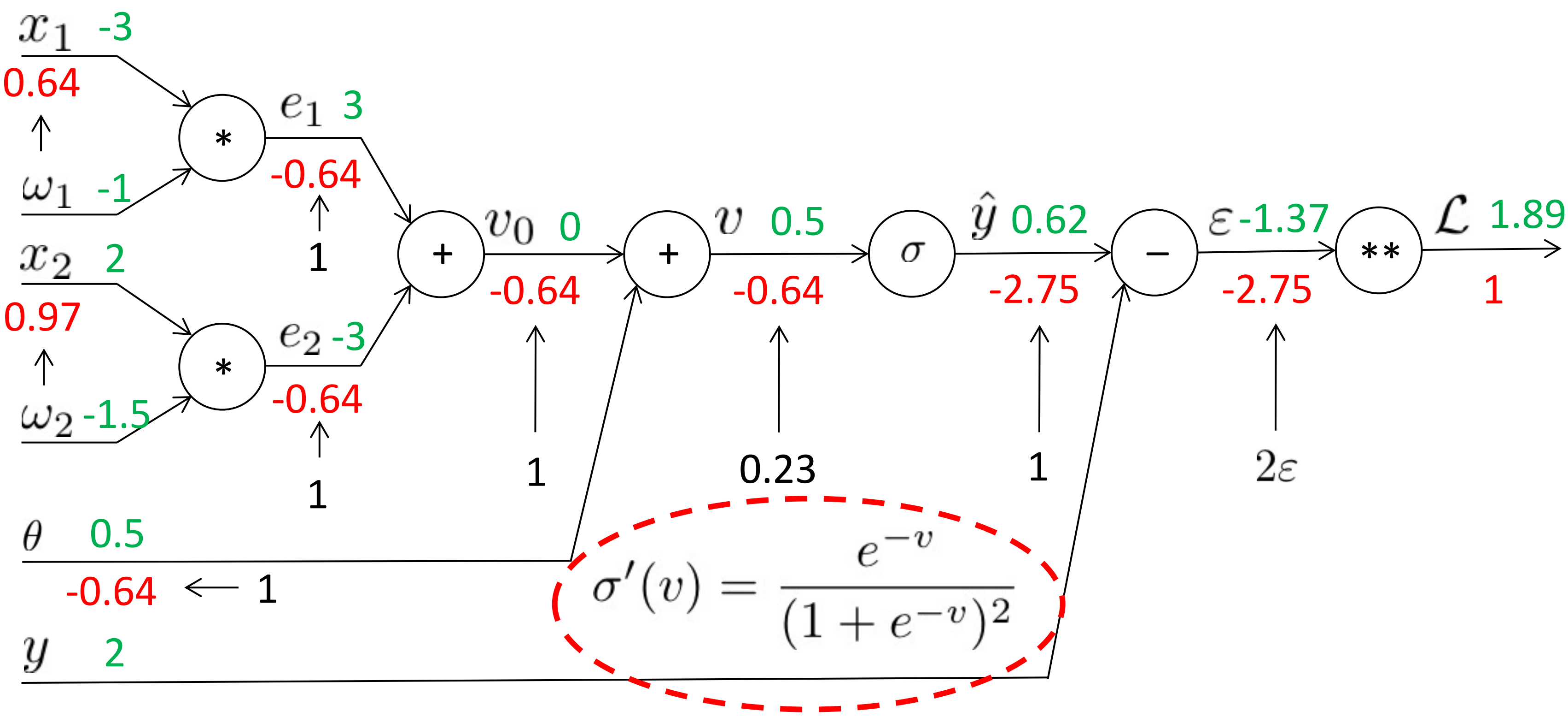
- Optimisation du calcul



> Propriété intéressante de la fonction sigmoïde pour son calcul

Calcul du gradient

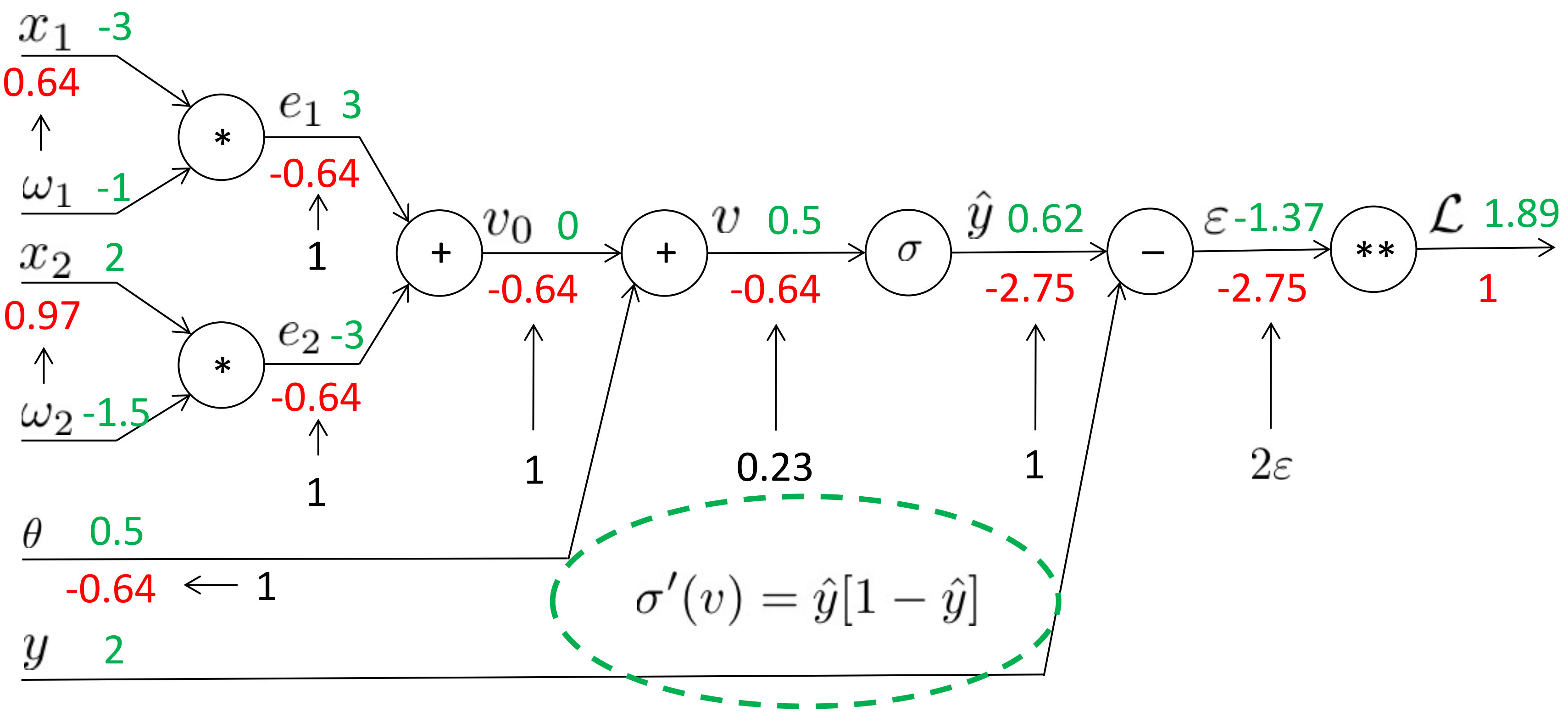
- Optimisation du calcul



$$\sigma'(v) = \frac{1+e^{-v}-1}{(1+e^{-v})^2} = \frac{1+e^{-v}}{(1+e^{-v})^2} - \frac{1}{(1+e^{-v})^2} = \sigma(v) - \sigma(v)^2 = \sigma(v)[1-\sigma(v)]$$

Calcul du gradient

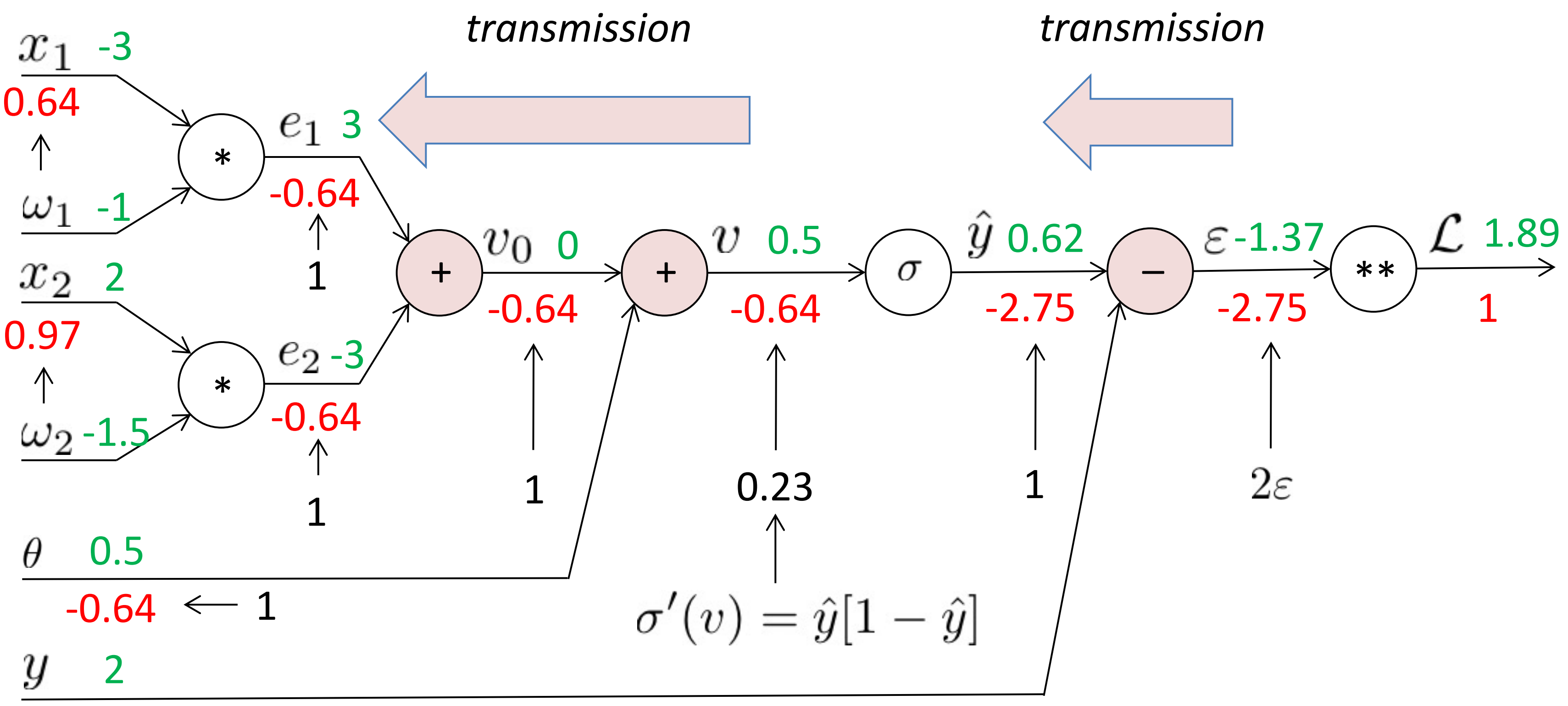
- Optimisation du calcul



> La sortie est déjà calculée!

Calcul du gradient

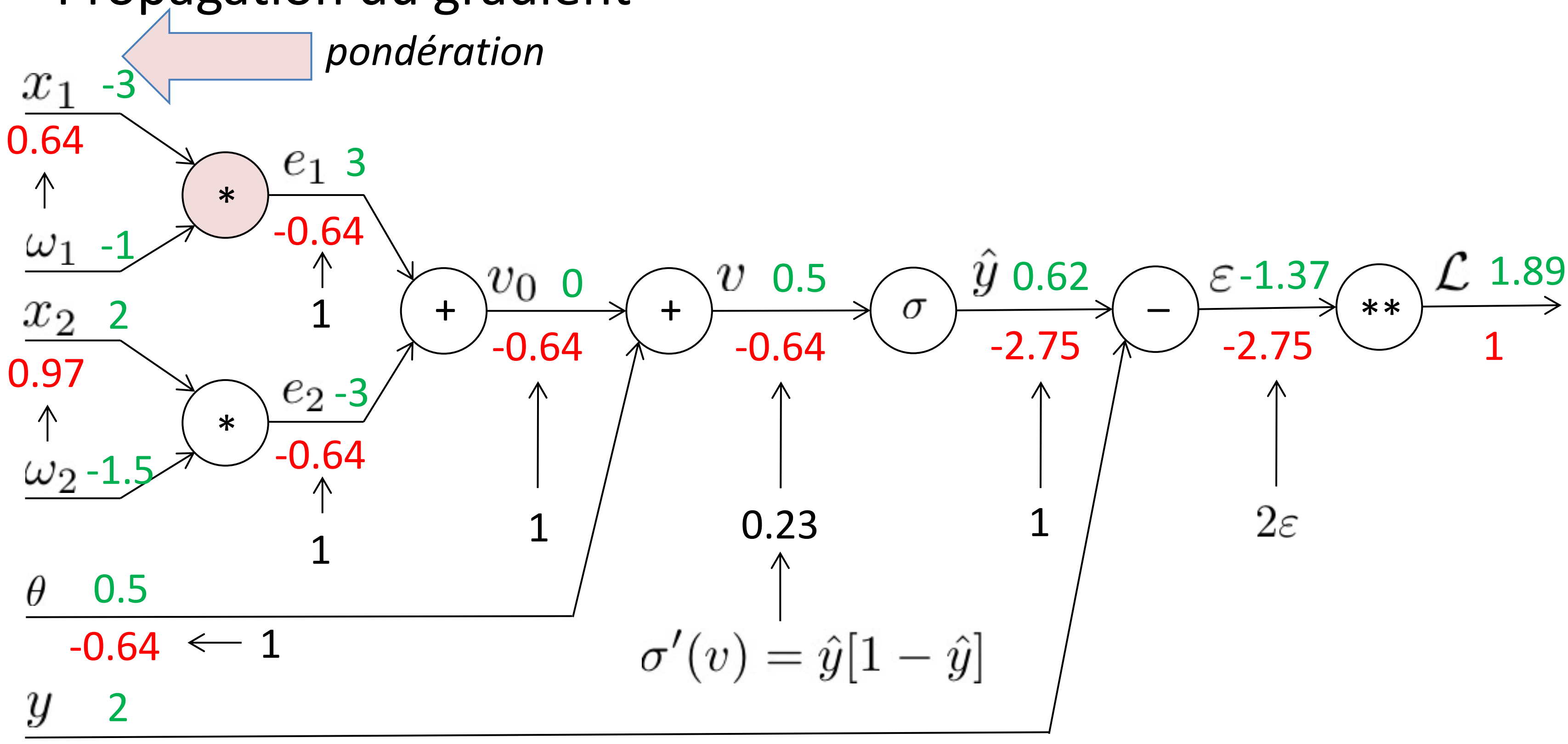
- Propagation du gradient



> Additions et soustractions laissent passer le gradient

Calcul du gradient

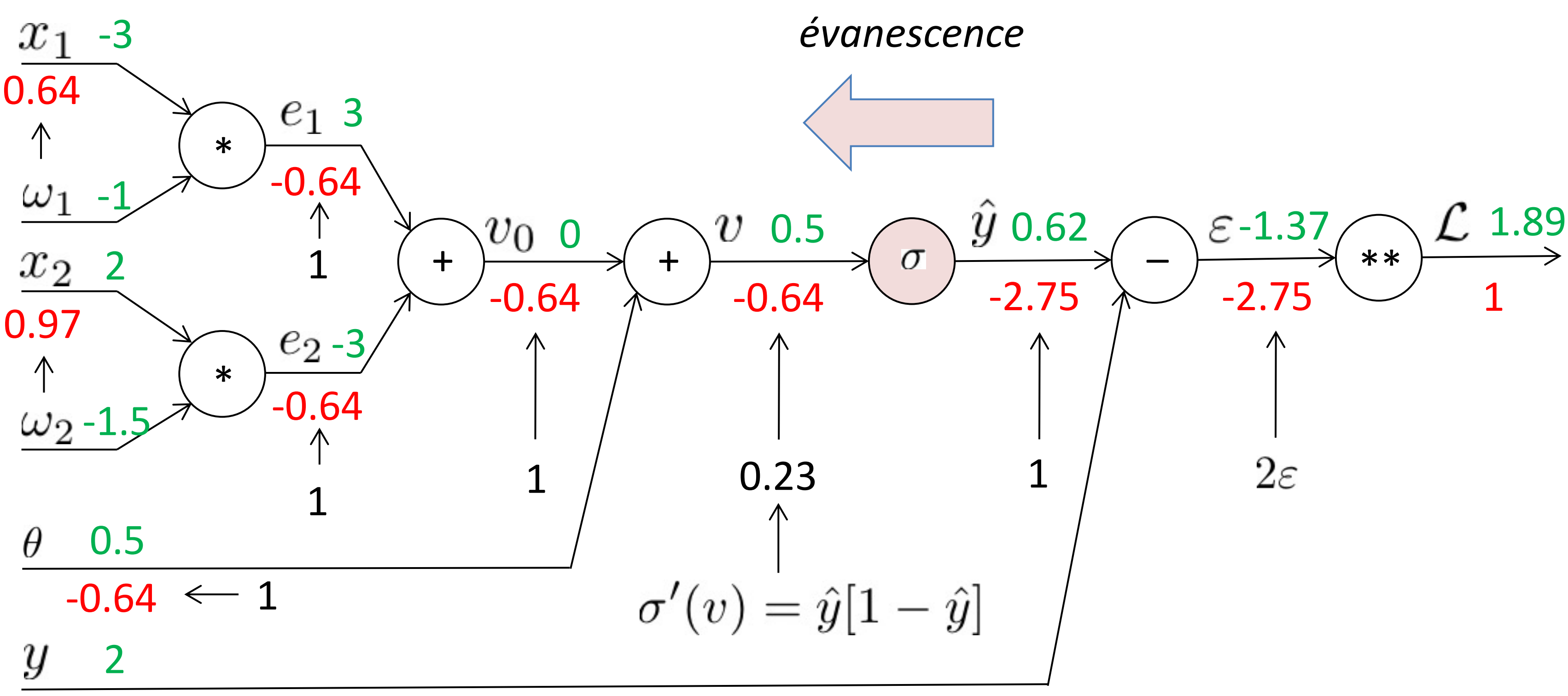
- Propagation du gradient



> Les multiplications pondèrent le gradient

Calcul du gradient

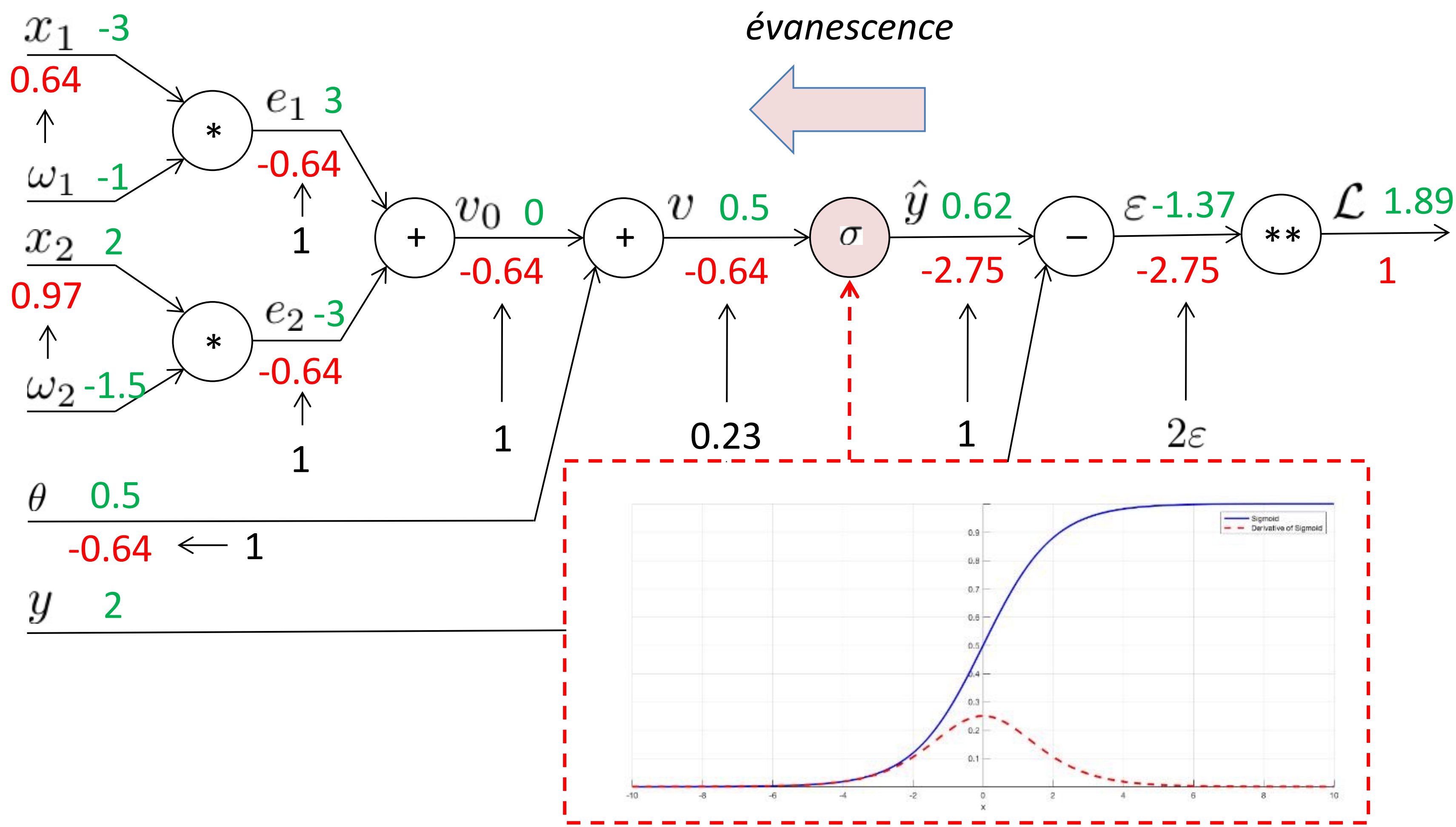
- Propagation du gradient



> La sigmoïde évanouit le gradient

Calcul du gradient

- Propagation du gradient

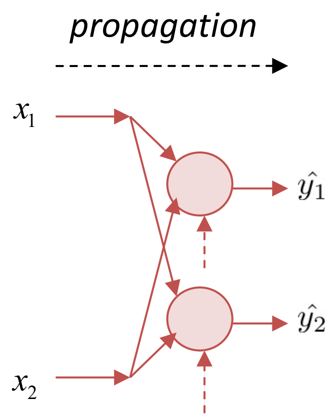


SCIENCES SORBONNE UNIVERSITÉ

5.
Perceptron multi-couches

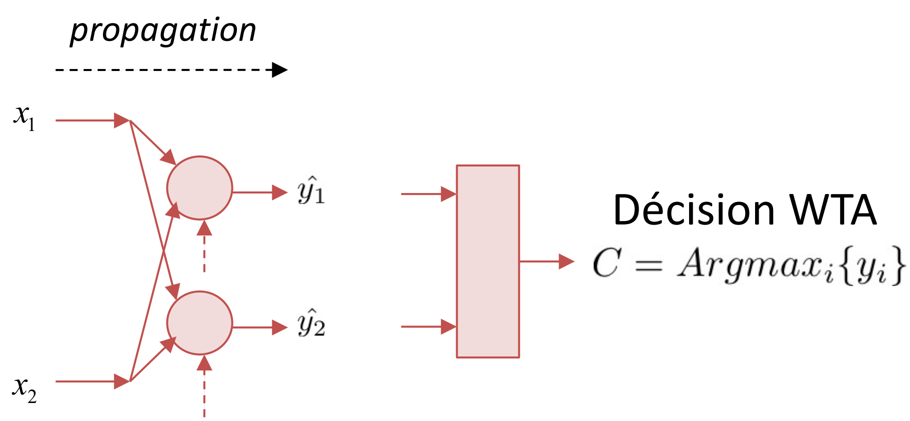
Perceptron multicouches

- Problèmes 2 classes à *n* classes
  - Propagation



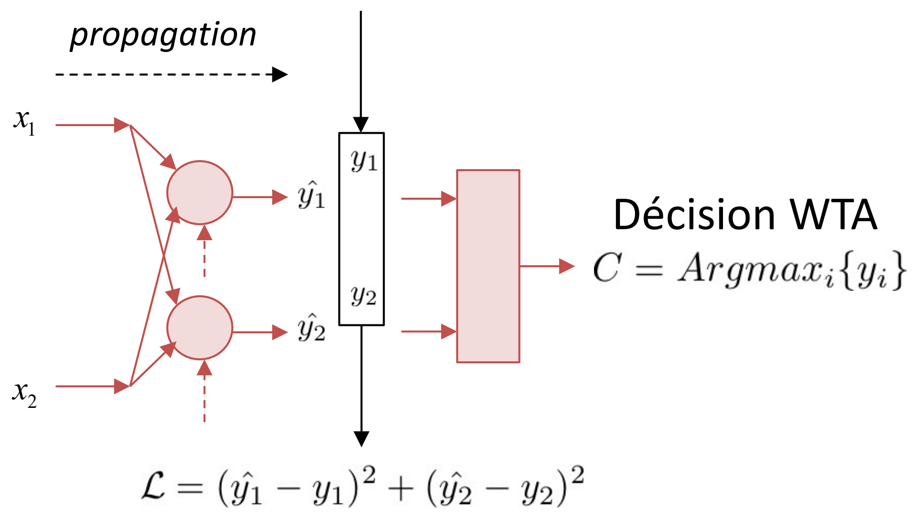
Perceptron multicouches

- Problèmes 2 classes à *n* classes
  - Propagation
  - Décision



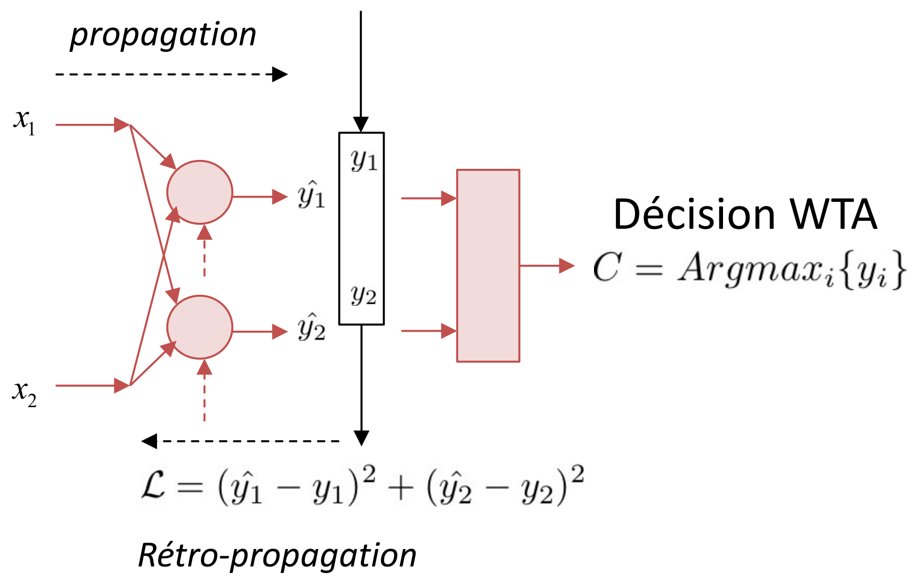
Perceptron multicouches

- Problèmes 2 classes à *n* classes
  - Propagation
  - Décision
  - Calcul de l'erreur



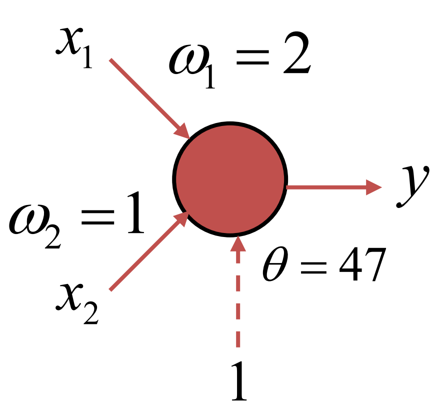
Perceptron multicouches

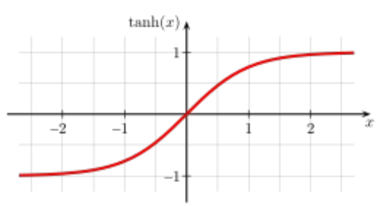
- Problèmes 2 classes à *n* classes
  - Propagation
  - Décision
  - Calcul de l'erreur
  - Rétro-propagation

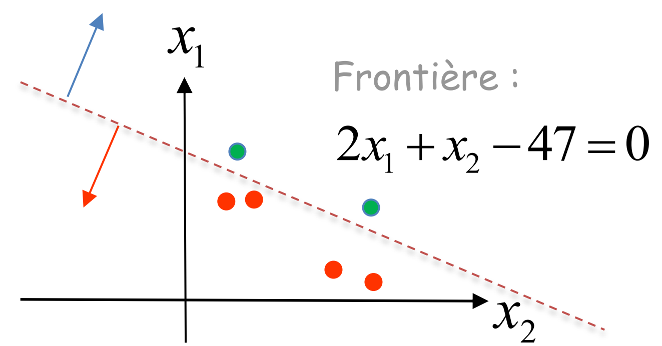


Perceptron multicouches

- Problèmes non linéairement séparables
  - Cas du neurone simple (avec la fonction de tangente-h)



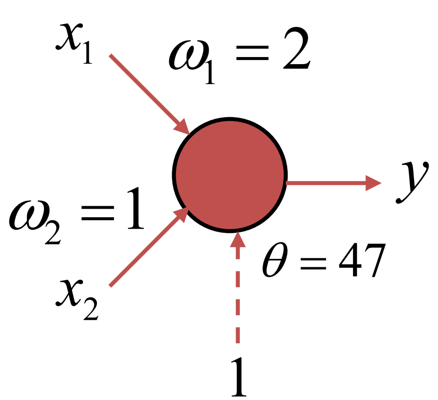


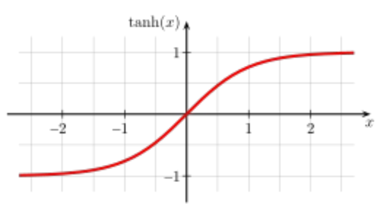


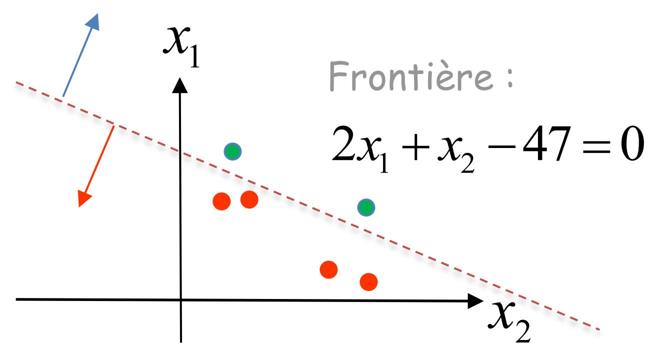
Perceptron multicouches

- Problèmes non linéairement séparables
  - Cas du neurone simple (avec la fonction de tangente-h)
  - Equation de la frontière

$$y = 0 \Leftrightarrow \sigma(\sum_i \omega_i x_i + \theta) = 0 \Leftrightarrow \boxed{\sum_i \omega_i x_i + \theta = 0}$$

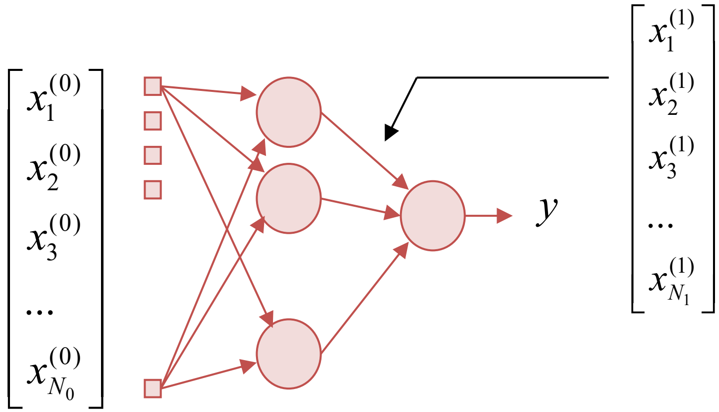






Perceptron multicouches

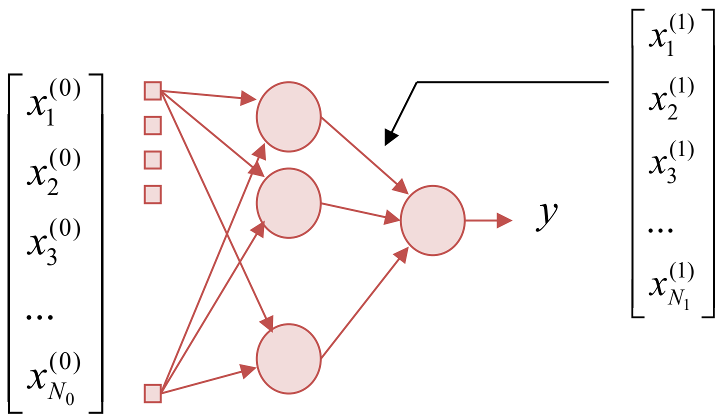
- Problèmes non linéairement séparables
  - Architecture 1 couche cachée (avec la fonction de tangente-h)



Perceptron multicouches

- Problèmes non linéairement séparables
  - Architecture 1 couche cachée (avec la fonction de tangente-h)
  - Equation de la frontière

$$y=\sigma\left(\sum_{j=0}^{N_1}\omega_j^{(2)}x_j^{(1)}\right)=0 \Leftrightarrow \sum_{j=0}^{N_1}\omega_j^{(2)}x_j^{(1)}=0 \Leftrightarrow \boxed{\sum_{j=0}^{N_1}\omega_j^{(2)}\left[\sigma\left(\sum_{l=0}^{N_0}\omega_{jl}^{(1)}x_l^{(0)}\right)\right]x_j^{1}=0}$$

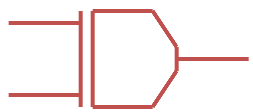


Perceptron multicouches

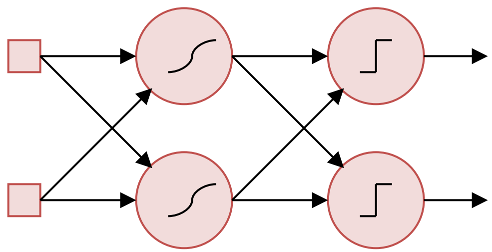
- Exemple du OU-Exclusif

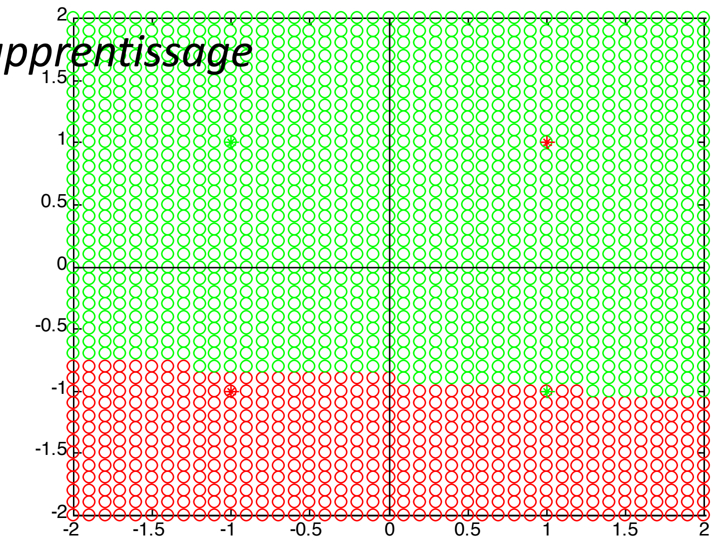
Avant apprentissage

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



s



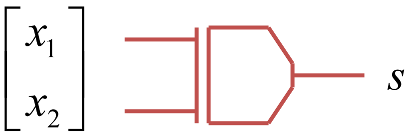


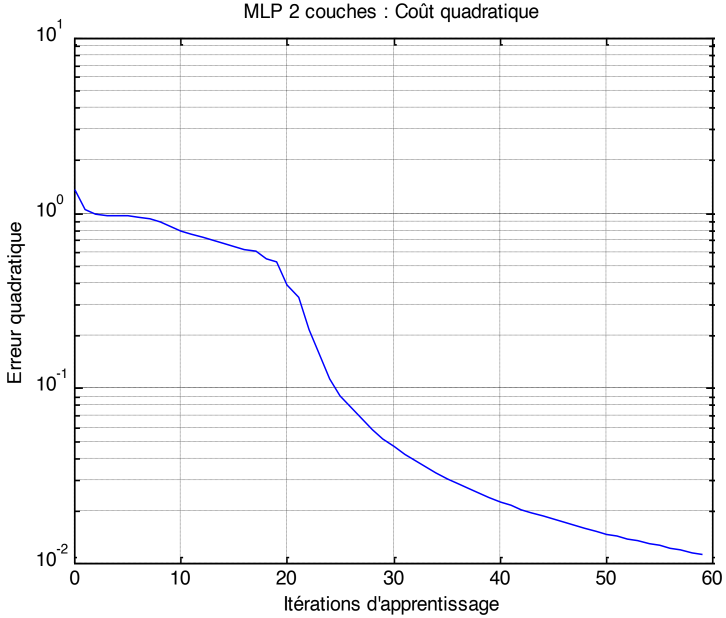
$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} +1 \\ -1 \end{bmatrix} \Rightarrow s = 1 \text{ (vert)}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \end{bmatrix} \Rightarrow s = 0 \text{ (rouge)}$$

Perceptron multicouches

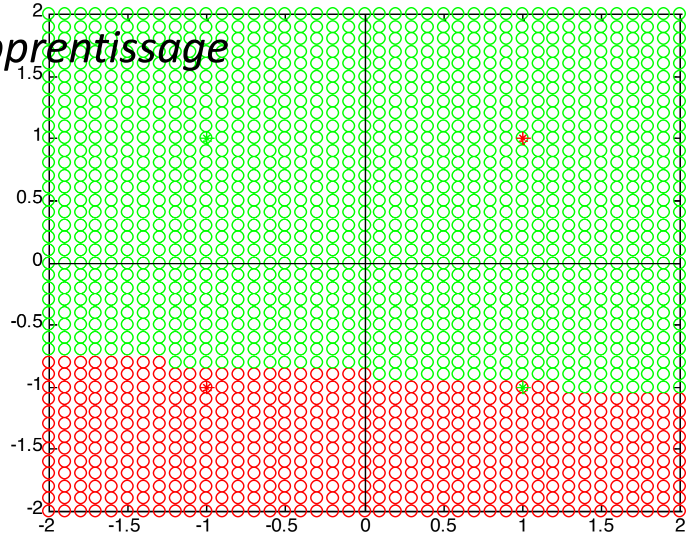
- Exemple du OU-Exclusif



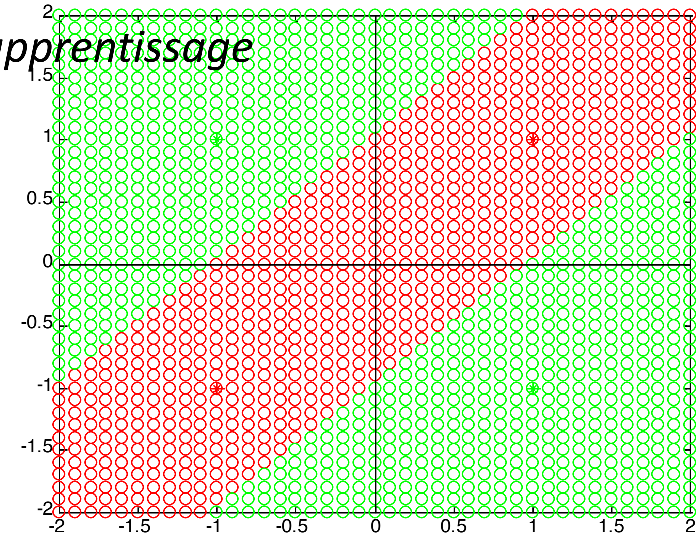


Evolution du coût quadratique

Avant apprentissage

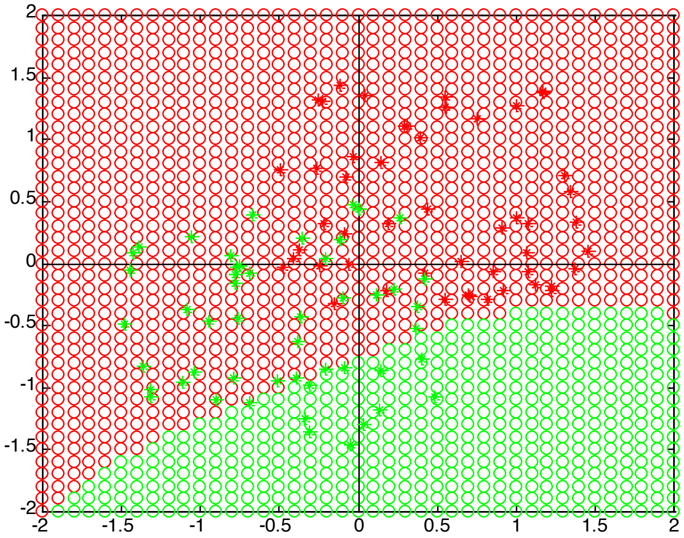


Après apprentissage

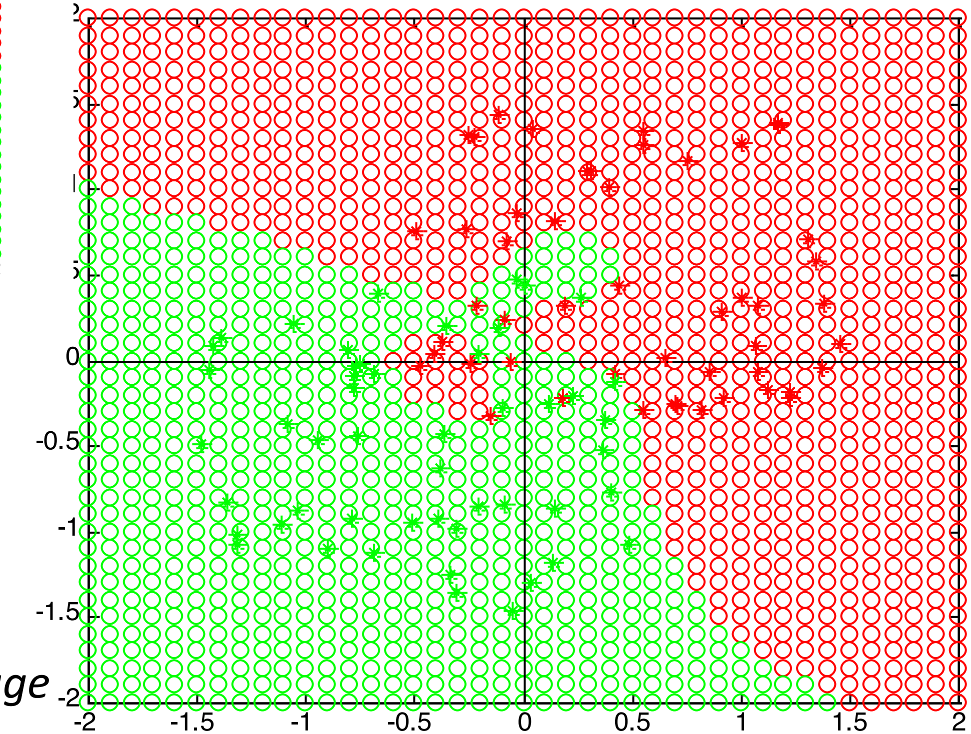


Perceptron multicouches

- Exemple nuages de points



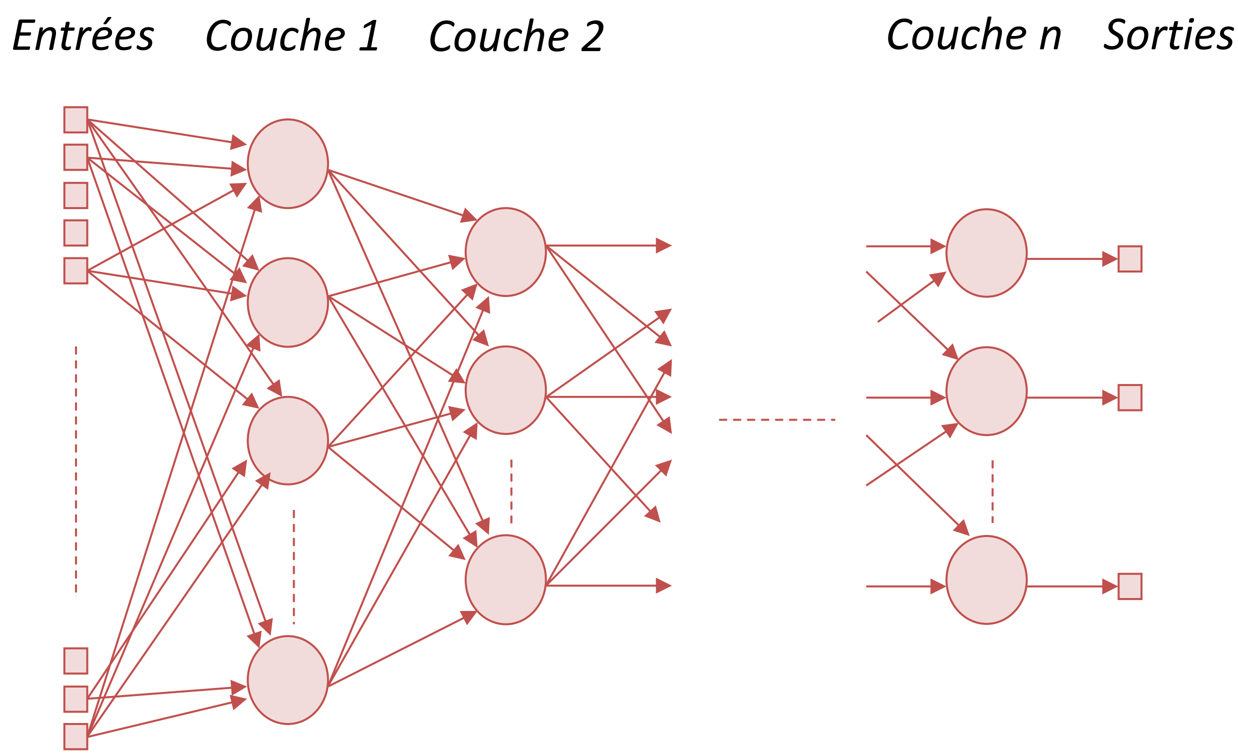
Avant apprentissage



Après apprentissage

Perceptron multicouches

- Structure générale
  - Connectivité totale d'une couche à la suivante (feedforward)
  - Pas de connexions récurrentes



Perceptron multicouches

- Structure générale
  - Connectivité totale d'une couche à la suivante (feedforward)
  - Pas de connexions récurrentes
  - Calcul matriciel (layer)
    - 1 neurone X *n* entrées

$$v = \sum_{i=1}^{n} \omega_i x_i$$

Perceptron multicouches

- Structure générale
  - Connectivité totale d'une couche à la suivante (feedforward)
  - Pas de connexions récurrentes
  - Calcul matriciel (layer)
    - 1 neurone X *n* entrées

$$v = [\omega_1, \omega_2, \ldots, \omega_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Perceptron multicouches

- Structure générale
  - Connectivité totale d'une couche à la suivante (feedforward)
  - Pas de connexions récurrentes
  - Calcul matriciel (layer)
    - 1 neurone X *n* entrées
    - *m* neurones X *n* entrées

$$v = [\omega_1, \omega_2, \ldots, \omega_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} \omega_{11}, \omega_{12}, \ldots, \omega_{1n} \\ \omega_{21}, \omega_{22}, \ldots, \omega_{2n} \\ \ldots \\ \omega_{m1}, \omega_{m2}, \ldots, \omega_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Perceptron multicouches

- Structure générale
  - Connectivité totale d'une couche à la suivante (feedforward)
  - Pas de connexions récurrentes
  - Calcul matriciel (layer)
    - 1 neurone X *n* entrées
    - *m* neurones X *n* entrées
    - *k* exemples X *m* neurones X *n* entrées

$$v = [\omega_1, \omega_2, \dots, \omega_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} \omega_{11}, \omega_{12}, \dots, \omega_{1n} \\ \omega_{21}, \omega_{22}, \dots, \omega_{2n} \\ \dots \\ \omega_{m1}, \omega_{m2}, \dots, \omega_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

$$\begin{bmatrix} v_1^1 & v_1^2 & & v_1^k \\ v_2^1 & v_2^2 & & v_2^k \\ \vdots & \vdots & \dots & \vdots \\ v_m^1 & v_m^2 & & v_m^k \end{bmatrix} = \begin{bmatrix} \omega_{11}, \omega_{12}, \dots, \omega_{1n} \\ \omega_{21}, \omega_{22}, \dots, \omega_{2n} \\ \dots \\ \omega_{m1}, \omega_{m2}, \dots, \omega_{mn} \end{bmatrix} \begin{bmatrix} x_1^1 & x_1^2 & & x_1^k \\ x_2^1 & x_2^2 & & x_2^k \\ \vdots & \vdots & \dots & \vdots \\ x_m^1 & x_m^2 & & x_m^k \end{bmatrix}$$

Rétropropagation du gradient

D.E. Rumelhart et al, 1986.

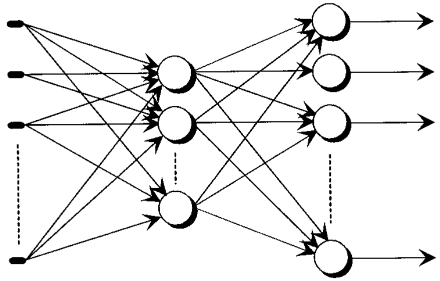
-*Learning representations by back-propagating errors* , Nature, vol. 323 (1986) 323.

Y. Le Cun, 1986.

-*Learning process in an assymetric threshold Network.* , Disordered systems and biological organizations. Nato-ASI Series ed. Bienenstock et al. Springer Verlag.

Dans ces articles sont décrits l'algorithme d'apprentissage dit de **rétropropagation de l'erreur** qui a fourni le moyen d'entraîner les réseaux de type Perceptron munis d'un nombre quelconque de **couches cachées**.

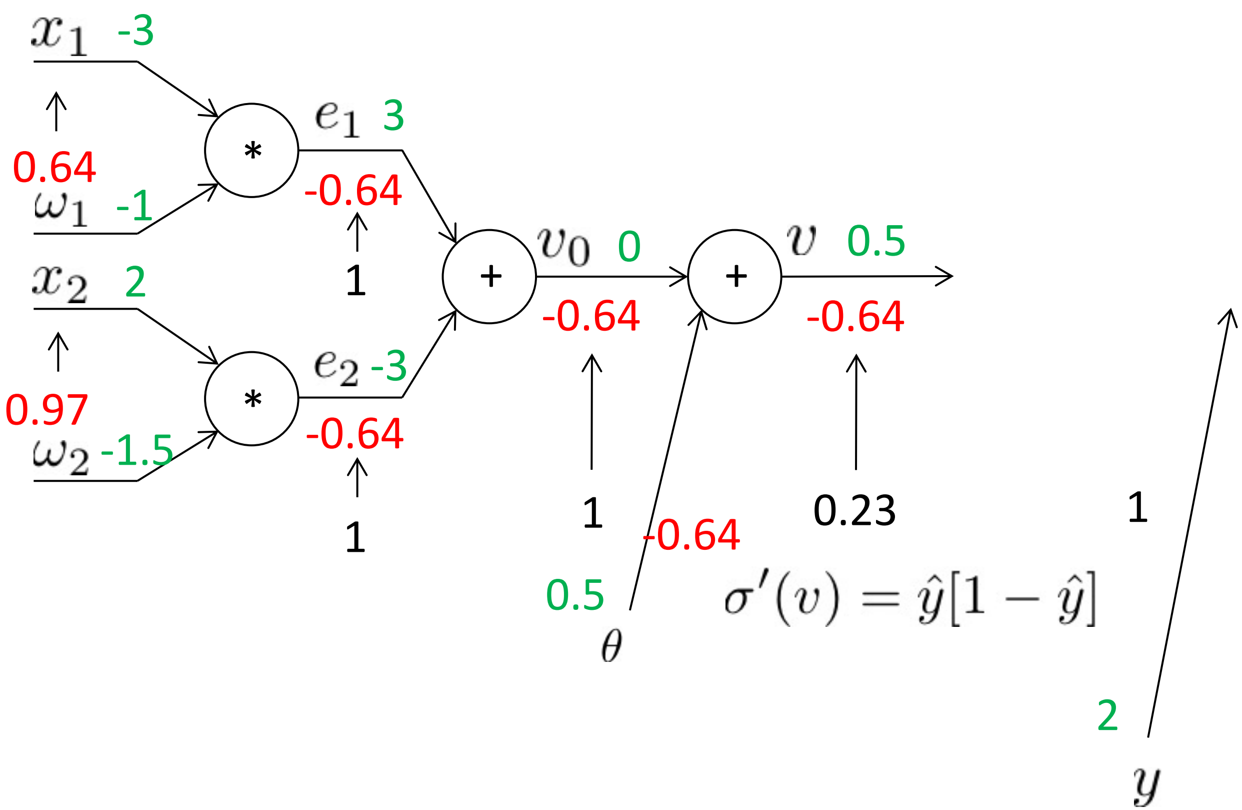
Cet algorithme a relancé l'intérêt pour les réseaux MLP (Multi-Layer Perceptron) au milieu des années 80.



6. Apprentissage

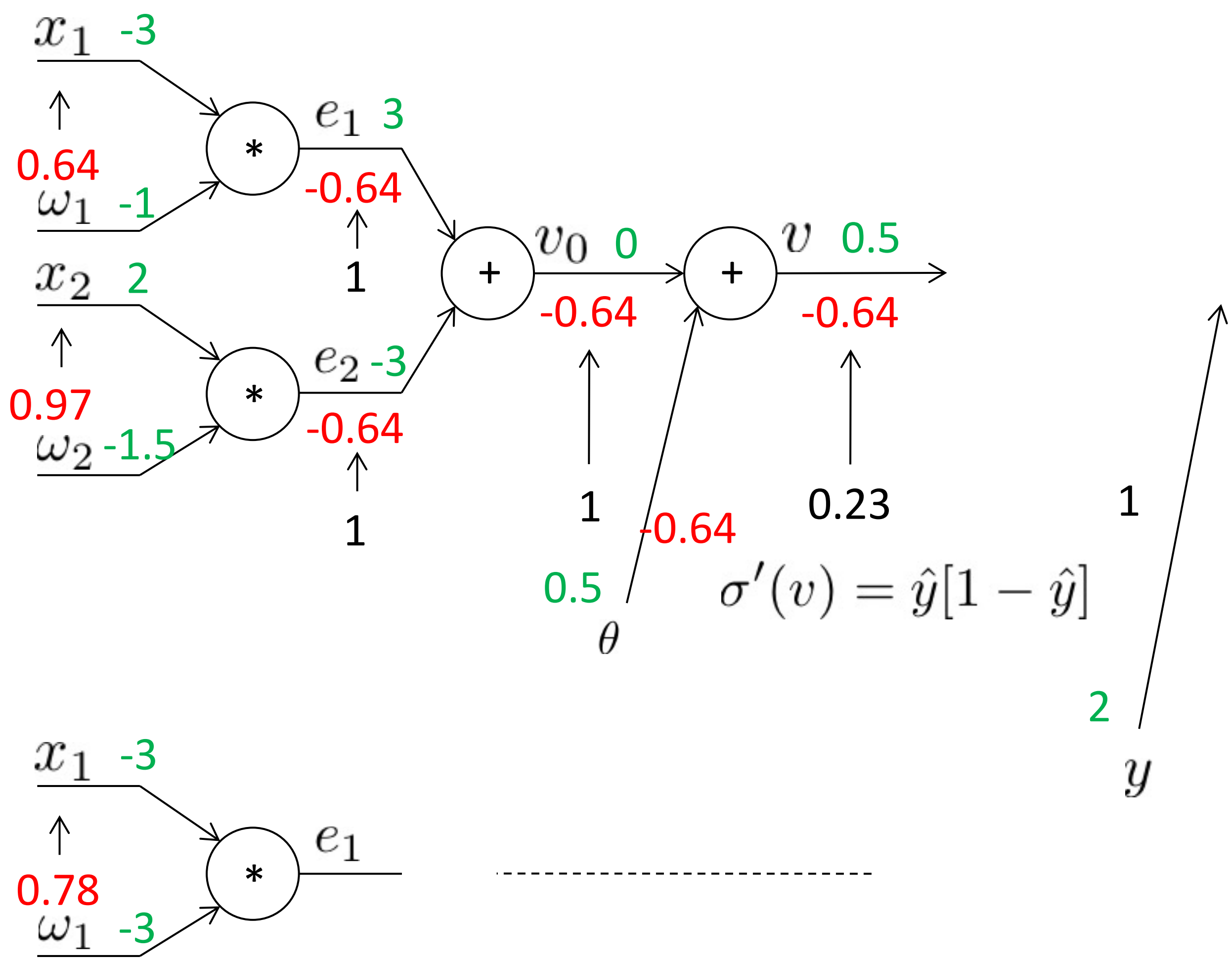
Rétropropagation du gradient

- Le calcul



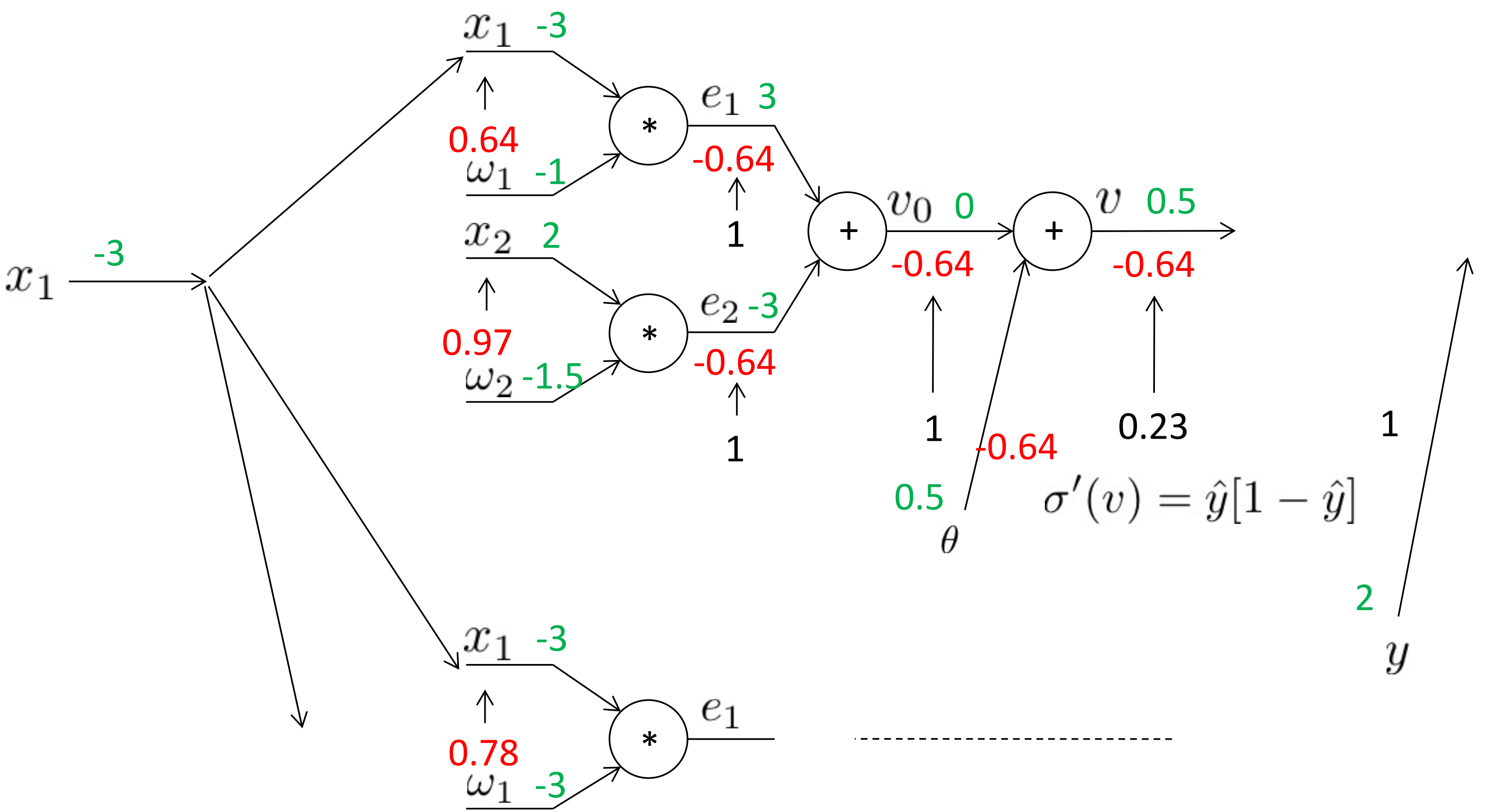
Rétropropagation du gradient

- Le calcul



Rétropropagation du gradient

- Le calcul



Rétropropagation du gradient

- Le calcul